



## LẬP TRÌNH ANDROID

# Module 2

☞ Click vào phụ lục để chuyển tới bài cần đọc

### Phụ lục

Bài 1 Hello Android.....	2
Bài 2 Các thành phần ứng dụng .....	14
Bài 3 View & Các điều khiển cơ bản .....	26
Bài 4 Tài nguyên ứng dụng cơ bản .....	39
Bài 5 Intent .....	58
Bài 6 Tài nguyên ứng dụng hình ảnh & giao diện .....	70
Bài 7 Asset – SharedPreference – Bộ nhớ thiết bị.....	88
Bài 8 Adapter & AdapterView .....	99
Bài 9 Fragment .....	110



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

# Lập trình Android

## Bài 1. Hello Android

*Ngành Mạng & Thiết bị di động*

cuu duong than cong . com 2014 



cuu duong than cong . com

## Nội dung



- 1. Tổng quan**
  - Hệ điều hành Android
  - Android và hệ sinh thái
- 2. Kiến trúc Android**
- 3. Môi trường phát triển ứng dụng Android**
- 4. Tạo ứng dụng đầu tiên**



## 1.1 Hệ điều hành Android



### Lịch sử phát triển

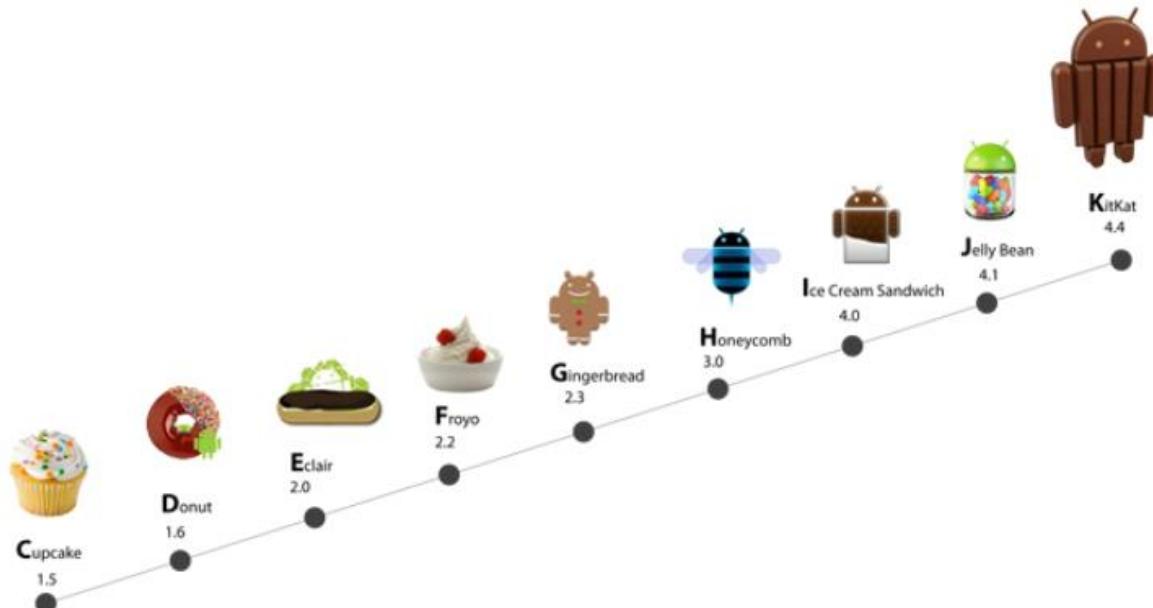
- Năm 2003, Android Inc. được thành lập bởi Andy Rubin, Rich Miner, Nick Sears và Chris White tại California.
- Năm 2005, Google sở hữu Android cùng với các vị trí quản lý.
- Năm 2007, OHA (Open Handset Alliance) được thành lập bởi Google cùng với nhiều nhà sản xuất thiết bị phần cứng, thiết bị không dây và vi xử lý. Công bố nền tảng phát triển Android.
- Năm 2008, thiết bị HTC Dream là phiên bản thử nghiệm đầu tiên hoạt động với hệ điều hành Android 1.0.
- Năm 2010, Google khởi đầu dòng thiết bị Nexus với thiết bị đầu tiên của HTC là Nexus One.
- Năm 2013, ra mắt loạt thiết bị phiên bản GPE.
- Năm 2014, Google công bố Android Wear, hệ điều hành dành cho các thiết bị đeo được.



## 1.1 Hệ điều hành Android



### ❑ Các phiên bản hệ điều hành



## 1.1 Hệ điều hành Android



### ❑ Các phiên bản hệ điều hành

- Phiên bản 1.x:
  - Android 1.0 (API 1)
  - Android 1.1 (API 2)
  - Android 1.5 Cupcake (API 3)
  - Android 1.6 Donut (API 4)
- Phiên bản 2.x:
  - Android 2.0 Eclair (API 5) – Android 2.0.1 (API 6) – Android 2.1 (API 7)
  - Android 2.2 – 2.2.3 Froyo (API 8)
  - Android 2.3 – 2.3.2 Gingerbread (API 9)
  - Android 2.3.3 – 2.3.7 Gingerbread (API 10)
- Phiên bản 3.x:
  - Android 3.0 Honeycomb (API 11)
  - Android 3.1 Honeycomb (API 12)
  - Android 3.2 Honeycomb (API 13)



## 1.1 Hệ điều hành Android



### ❑ Các phiên bản hệ điều hành

- Phiên bản 4.x:

- Android 4.0 – 4.0.2 Ice Cream Sandwich (API 14)
- Android 4.0.3 – 4.0.4 Ice Cream Sandwich (API 15)
- Android 4.1 Jelly Bean (API 16)
- Android 4.2 Jelly Bean (API 17)
- Android 4.3 Jelly Bean (API 18)
- Android 4.4 Kit Kat (API 19)



## 1.2 Android và hệ sinh thái



### ❑ Hệ sinh thái





## 1.2 Android và hệ sinh thái

### □ Hệ sinh thái



cuuduongthancong . com

8



## Nội dung

1. Tổng quan
2. Kiến trúc Android
  - Cấu tạo
  - Kiến trúc phần mềm
  - Ngôn ngữ lập trình
3. Môi trường phát triển ứng dụng Android
4. Tạo ứng dụng đầu tiên



Lập trình Android (2014) - Bài 1. Hello Android

9

## 2.1 Cấu tạo

- ❑ **Android được hình thành dựa trên nền tảng Linux nhân 2.6, từ phiên bản 4.0 sử dụng Linux nhân 3.x.**
- ❑ **Android bao gồm 3 thành phần chính:**
  - Middleware
  - Các thư viện và API viết bằng C
  - Các ứng dụng thực thi viết bằng Java
- ❑ **Sử dụng máy ảo Dalvik để biên dịch mã .dex (Dalvik Executable) sang Java bytecode.**



## 2.2 Kiến trúc





## 2.3 Ngôn ngữ lập trình

❑ Có thể sử dụng các ngôn ngữ lập trình:

- Java
- C/C++
- JNI
- XML
- Render Script



## Nội dung

1. Tổng quan
2. Kiến trúc Android
3. Môi trường phát triển ứng dụng Android
  - Android Developer Tools Bundle
  - Android Studio
4. Tạo ứng dụng đầu tiên



### 3.1 Android Developer Tools Bundle

❑ **Android Developer Tools Bundle (ADT) bao gồm:**

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform Tools
- Phiên bản hệ điều hành Android
- Tập tin cài đặt hệ điều hành cho máy ảo



### 3.2 Android Studio

❑ **Android Studio là công cụ lập trình dựa trên nền IntelliJ, cung cấp các tính năng mạnh mẽ hơn ADT, bao gồm:**

- Hỗ trợ xây dựng dự án dạng Gradle.
- Hỗ trợ sửa lỗi nhanh và tái sử dụng cấu trúc phương thức
- Cung cấp các công cụ kiểm tra tính khả dụng, khả năng hoạt động của ứng dụng, tương thích nền tảng...
- Hỗ trợ bảo mật mã nguồn và đóng gói ứng dụng.
- Trình biên tập giao diện cung cấp tổng quan giao diện ứng dụng và các thành phần, cho phép tùy chỉnh trên nhiều cấu hình khác nhau.
- Cho phép tương tác với nền Google Cloud.



## Nội dung



1. Tổng quan
2. Kiến trúc Android
3. Môi trường phát triển ứng dụng Android
4. Tạo ứng dụng đầu tiên
  - Khởi tạo dự án
  - Cấu trúc dự án
  - AndroidManifest
  - Tạo máy ảo và cài đặt ứng dụng



### 4.1 Khởi tạo dự án



- ❑ **Khởi chạy Eclipse, tiến hành chọn nơi lưu trữ dự án, sau đó thực hiện các bước sau:**
- Trên thanh menu, chọn File → New → Android Application Project
  - Tiến hành đặt tên ứng dụng, tên dự án, tên nhà phát triển và chọn phiên bản Android muốn phát triển.
  - Tuỳ chỉnh biểu tượng ứng dụng.
  - Nhấn Finish để hoàn thành tạo dự án.



## 4.2 Cấu trúc dự án



### □ Cấu trúc dự án có thể chia thành phần sau:

- Thư mục:
  - src: chứa mã nguồn
  - gen: chứa mã nguồn tự phát sinh
  - Android x.x – Android Private Library: cung cấp API dựa trên phiên bản hệ điều hành phát triển.
  - asset: chứa tập tin tài nguyên không biên dịch.
  - bin: chứa các tập tin đóng gói.
  - res: chứa các tập tin và thư mục tài nguyên.
- Tập tin:
  - AndroidManifest.xml: chứa thông tin cài đặt ứng dụng
  - Proguard-project.txt: chuyên quyền về chế độ bảo mật
  - Project.properties: chứa thông tin về dự án



## 4.3 AndroidManifest



### □ Vai trò của tập tin AndroidManifest.xml:

- Lưu trữ thông tin tên gói ứng dụng, tồn tại duy nhất một tên gói cho mỗi ứng dụng.
  - Ví dụ: com.htsi.myfirstapp
- Cho biết ứng dụng sử dụng các thành phần nào, mỗi thành phần được khai trong một cặp thẻ.
  - Ví dụ: <activity>....</activity>
- Định nghĩa tiến trình quản lý các thành phần ứng dụng.
- Định nghĩa các quyền sử dụng API và truy xuất ứng dụng khác.
- Qui định các yêu cầu khi được ứng dụng khác truy xuất.
- Khai báo cấp độ API tối thiểu xây dựng ứng dụng.
- Khai báo các thư viện có liên quan.



## 4.4 Tạo máy ảo và cài đặt ứng dụng



### ❑ Tạo máy ảo:

- Khởi chạy Android Virtual Machine Manager.
- Chọn New và điền các thông tin cần thiết:
  - AVD Name: tên máy ảo
  - Device: chủng loại thiết bị
  - Target: phiên bản hệ điều hành
  - CPU/ABI: loại vi xử lý
  - Keyboard: sử dụng bàn phím từ PC
  - Skin: kích thước màn hình
  - Front Camera/ Back Camera: máy ảnh
  - Memory Options: vùng nhớ RAM và HEAP
  - Internal Storage: bộ nhớ trong thiết bị
  - SDCard: bộ nhớ trên thẻ nhớ
  - Emulation Options: giả lập bộ xử lý đồ họa
- Nhấn OK để hoàn thành thiết lập
- Chọn máy ảo vừa tạo → Start.

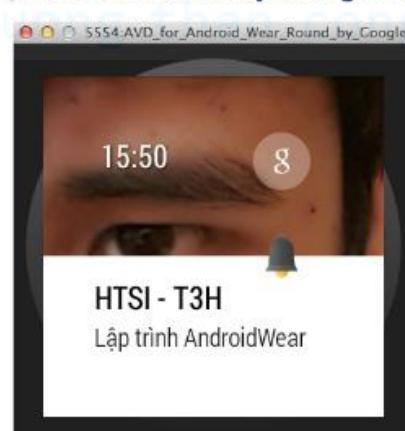


## 4.4 Tạo máy ảo và cài đặt ứng dụng



### ❑ Cài đặt ứng dụng:

- Chọn dự án cần biên dịch và cài đặt.
- Nhấn chuột phải → Run As → Android Application.
- Chọn thiết bị cần cài đặt.
- Quan sát giao diện Console để thấy thông tin cài đặt.



## Thảo luận



cuu duong than cong . com

Lập trình Android (2014) - Bài 1. Hello Android

22

cuu duong than cong . com



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

## Lập trình Android

### Bài 2. Các thành phần ứng dụng

*Ngành Mạng & Thiết bị di động*

cuu duong than cong . com

2014



5014



cuu duong than cong . com

## Nội dung



### 1. Các thành phần ứng dụng

- Activity
- View
- Service
- Broadcast Receiver
- Intent
- Content Provider
- Notification

### 2. Ứng dụng Android và cơ chế hoạt động

### 3. Activity và vòng đời ứng dụng



### 1.1 Activity



- ❑ Trong ứng dụng Android, Activity đóng vai trò là một màn hình, nơi người dùng có thể tương tác với ứng dụng, ví dụ: chụp hình, xem bản đồ, gửi mail...
- ❑ Một ứng dụng có thể có một hoặc nhiều Activity, Activity được khởi chạy đầu tiên khi ứng dụng hoạt động được gọi là “MainActivity”.
- ❑ Activity có thể hiển thị ở chế toàn màn hình, hoặc ở dạng cửa sổ với một kích thước nhất định.
- ❑ Các Activity có thể gọi đến các Activity khác, Activity được gọi sẽ nhận được tương tác ở thời điểm đó.



## 1.2 View

- ❑ View được sử dụng để tạo ra các điều khiển trên màn hình cho phép nhận các tương tác từ người dùng cũng như hiển thị các thông tin cần thiết.
- ❑ View bao gồm hai dạng:
  - View: các điều khiển đơn lẻ
  - ViewGroup: tập hợp nhiều điều khiển đơn lẻ



## 1.3 Service

- ❑ Service được sử dụng để thực thi các tác vụ cần nhiều thời gian, thực hiện ở chế độ ngầm và thường không cần giao diện hiển thị.
- ❑ Service có thể được khởi chạy và hoạt động xuyên suốt ngay cả khi ứng dụng không hoạt động.
- ❑ Một số tác vụ cần thực hiện bằng Service:
  - Trình diễn các tập tin đa truyền thông như nhạc, phim...
  - Kết nối và thực hiện tải các nội dung thông qua Internet
  - Truy xuất đọc ghi tập tin



## 1.4 Broadcast Receiver



- ❑ Thành phần ứng dụng cho phép truyền tải các thông báo trên phạm vi toàn hệ thống. Không có giao diện nhưng có thể thực hiện thông báo qua thanh trạng thái.
- ❑ Broadcast Receiver truyền thông báo ở hai dạng:
  - Hệ thống: các thông báo được truyền trực tiếp từ hệ thống như: tắt màn hình, pin yếu, thay đổi kết nối...
  - Ứng dụng: xây dựng các truyền thông báo đến các thành phần trong ứng dụng như: khởi động Service, tải nội dung đến ứng dụng...



## 1.5 Intent



- ❑ Intent là đối tượng mang thông điệp, cho phép tạo ra các yêu cầu hành động giữa các thành phần trong ứng dụng, hoặc giữa các ứng dụng với nhau.
- ❑ Được sử dụng nhiều trong 3 trường hợp sau:
  - Khởi động Activity
  - Khởi động Service
  - Chuyển phát thông tin cho Broadcast Receiver



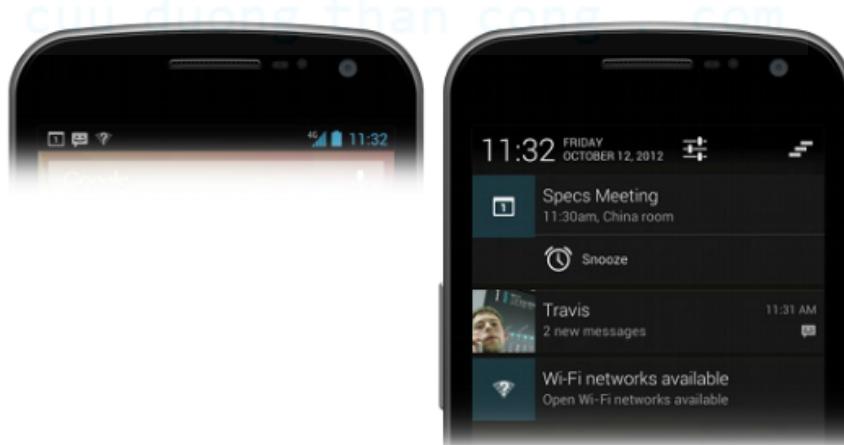
## 1.6 Content Provider

- ❑ Content Provider xây dựng cách thức truy xuất tập hợp các dữ liệu ứng dụng, dữ liệu có thể lưu trữ ở nhiều dạng như: SQLite, tập tin, tài nguyên Web hoặc bất kì thư mục lưu trữ nào.
- ❑ Có thể sử dụng Content Provider để xây dựng các ứng dụng sử dụng chung nguồn tài nguyên hoặc sử dụng riêng.
- ❑ Trong Android, một số Content Provider được xây dựng sẵn:
  - Danh bạ
  - Tài nguyên đa truyền thông
  - Lịch



## 1.7 Notification

- ❑ Notification được xây dựng cho mục đích gửi các thông báo đến người dùng thông qua thanh trạng thái.
- ❑ Giao diện Notification không thuộc giao diện ứng dụng, nhưng có thể tùy chỉnh giao diện Notification thông qua các phương thức có sẵn.



## Nội dung



1. Các thành phần ứng dụng
2. Ứng dụng Android và cơ chế hoạt động
  - Đóng gói và thực thi ứng dụng
  - Tính tương thích thiết bị
3. Activity và vòng đời ứng dụng



## 2.1 Đóng gói và thực thi ứng dụng



- ❑ **Ứng dụng Android được viết bằng ngôn ngữ Java và biên dịch, đóng gói cùng các tập tin tài nguyên thành tập tin \*.apk.**
- ❑ **Cài đặt trên thiết bị theo đường dẫn `data/app/<Tên đóng gói>`, được chứa trong **Sandbox** và được hiểu:**
  - Mỗi ứng dụng là một dạng “người dùng” khác nhau.
  - Mỗi ứng dụng được cấp một ID, do đó chỉ duy nhất ứng dụng mới có thể truy xuất các tập tin liên quan đến ứng dụng đó.
  - Ứng dụng thực thi riêng biệt trên từng máy ảo.
  - Tiết trình Linux được cấp phát khi bắt đầu thành phần ứng dụng được gọi thực thi, và thu hồi khi chấm dứt hoạt động.
  - Các ứng dụng có cùng ID và chứng chỉ (Certificate) có thể truy xuất tài nguyên của nhau, hoặc xin quyền nếu truy xuất hệ thống.





## 2.2 Tính tương thích thiết bị

### Tính tương thích ứng dụng với thiết bị bao gồm:

- Trang bị tính năng của thiết bị
- Phiên bản hệ điều hành
- Kích thước màn hình



## 2.2 Tính tương thích thiết bị

### Trang bị tính năng của thiết bị:

- Mỗi tính năng phần cứng và phần mềm trên thiết bị Android được cung cấp một ID, qui định thiết bị đó có được trang bị tính năng đó hay không.

#### ▪ Ví dụ:

- FEATURE\_SENSOR\_COMPASS: tính năng la bàn
- FEATURE\_APP\_WIDGET: tính năng gắn Widget





## 2.2 Tính tương thích thiết bị

### Phiên bản hệ điều hành:

- Mỗi phiên bản kế tiếp được bổ sung hoặc lược bỏ các hàm API, cần thực thi khai báo các thông tin phiên bản để sử dụng đầy đủ các tính năng cho ứng dụng.
  - Ví dụ:
    - Android ICS 4.0: bổ sung API cho Calendar
    - Android Kit Kat 4.4: bổ sung API cho WirelessPrint



## 2.2 Tính tương thích thiết bị

### Kích thước màn hình:

- Ứng dụng Android cần tương thích với nhiều kích cỡ thiết bị, được phân chia thành hai thuộc tính:
  - Kích thước vật lý màn hình.
  - Độ phân giải (mật độ điểm ảnh)



## Nội dung



1. Các thành phần ứng dụng
2. Ứng dụng Android và cơ chế hoạt động
3. Activity và độ ưu tiên ứng dụng
  - Xây dựng Activity
  - Quản lý trạng thái Activity
  - Độ ưu tiên trong ứng dụng



### 3.1 Xây dựng Activity



- Thực hiện tạo Activity cho ứng dụng:
  - Tạo mới lớp kế thừa từ lớp Activity
  - Thực thi các hàm quản lý trạng thái Activity
  - Xây dựng giao diện trong tài nguyên **res/layout**
  - Khai báo Activity trong tập tin **AndroidManifest.xml**



## 3.2 Quản lý trạng thái Activity



### ❑ Activity bao gồm ba trạng thái:

- Resumed: đang trong trạng thái nhận tương tác.
- Paused: không thể tương tác nhưng vẫn được thấy bởi người dùng.
- Stopped: thực hiện chạy ở chế độ ngầm.

### ❑ Thực hiện gọi các hàm quản lý trạng thái:

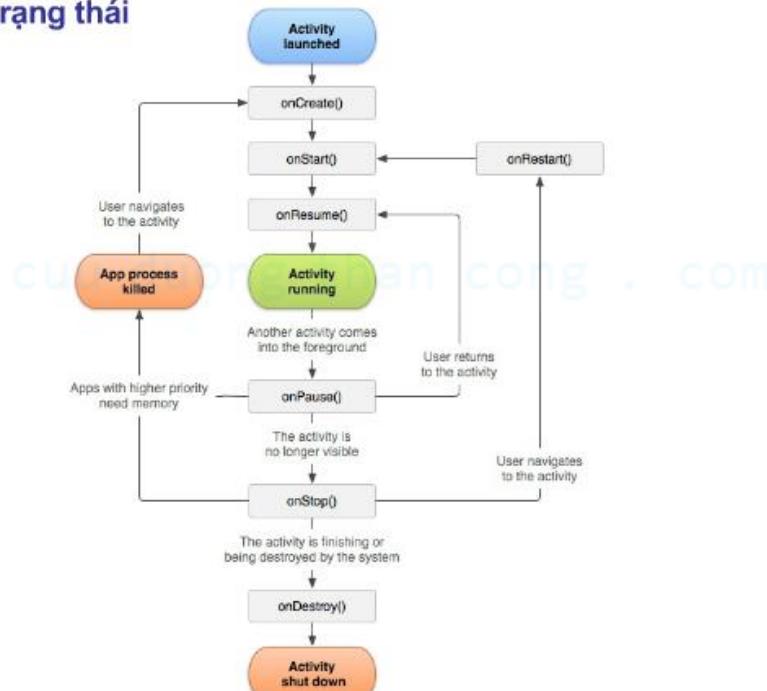
- onStart
- onRestart
- onCreate
- onPause
- onResume
- onStop
- onDestroy



## 3.2 Quản lý trạng thái Activity



### ❑ Sơ đồ trạng thái





### 3.3 Độ ưu tiên ứng dụng

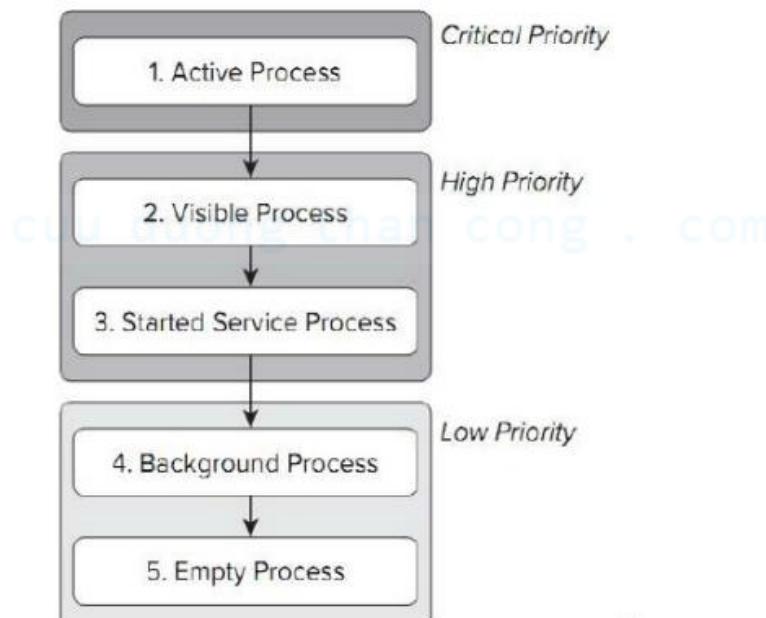
#### ❑ Cơ chế quản lý bộ nhớ:

- Android quản lý các Ứng dụng dựa trên độ ưu tiên.
- Nếu hai ứng dụng có cùng trạng thái thì ứng dụng nào đã chạy lâu hơn sẽ có *độ ưu tiên thấp hơn*.
- Nếu ứng dụng đang chạy một Service hay Content Provider do một ứng dụng khác hỗ trợ thì sẽ có cùng độ ưu tiên với ứng dụng đó.
- Các ứng dụng sẽ bị đóng mà không có sự báo trước.



### 3.3 Độ ưu tiên ứng dụng

#### ❑ Độ ưu tiên:



## Thảo luận





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

## Lập trình Android

### Bài 3. View & Các điều khiển cơ bản

*Ngành Mạng & Thiết bị di động*

cuu duong than cong . com

2014



5014



cuu duong than cong . com

## Nội dung



### 1. Khái niệm View

- View & ViewGroup
- Thể hiện & Thao tác
- Thuộc tính

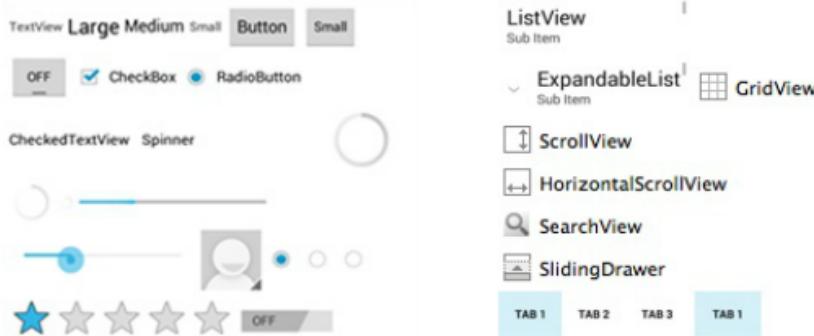
### 2. Các điều khiển cơ bản



### 1.1 View & ViewGroup



- View được sử dụng để tạo ra các điều khiển trên màn hình cho phép nhận các tương tác từ người dùng cũng như hiển thị các thông tin cần thiết.
- View bao gồm hai dạng:
  - View: các điều khiển đơn lẻ
  - ViewGroup: tập hợp nhiều điều khiển đơn lẻ



## 1.2 Thẻ hiện và Thao tác



### ❑ Thẻ hiện:

- Các đối tượng View được thể hiện trên màn hình giao diện như một hình chữ nhật tùy thuộc vị trí, kích thước, màu sắc và nhận vào cũng như xử lý các tương tác có liên quan.
- Một số thẻ hiện của lớp View: TextView, ImageView, SurfaceView...
- ViewGroup cũng là một thẻ hiện của View.

### ❑ Có thể xây dựng đối tượng View theo 2 cách:

- Kéo thả và tuỳ chỉnh thuộc tính trong XML.
- Thiết lập thông số và truy xuất trực tiếp trong Java Code.



## 1.2 Thẻ hiện và Thao tác



### ❑ Thao tác:

- Các đối tượng View được xây dựng và thiết lập với bốn thao tác chính:
  - Hiển thị nội dung thông qua phương thức `set<TT>(TS)`.
    - Ví dụ: TextView hiển thị văn bản, ImageView hiển thị hình ảnh...
  - Yêu cầu tương tác
    - Ví dụ: sử dụng `requestFocus` để yêu cầu tương tác với điều khiển.
  - Thiết lập chế độ hiển thị thông qua phương thức `setVisibility` (hoặc thuộc tính `visibility`: trong XML)
    - `VISIBLE`
    - `INVISIBLE`
    - `GONE`
  - Xây dựng phương thức "lắng nghe"
    - Ví dụ: bắt lại các sự kiện xảy ra trên điều khiển.



## 1.2 Thẻ hiện và Thao tác



### □ Thao tác:

- Một số sự kiện trên đối tượng View:

- OnClickListener
  - OnTouchListener
  - OnLongClickListener
  - OnDragListener
  - OnKeyListener

- Ví dụ: lắng nghe sự kiện nhấn

```
view.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Log.i("HTSI", "onClick");  
    }  
});
```



## 1.3 Thuộc tính



### □ Id:

- Khai báo kiểu số nguyên int, đánh dấu vùng nhớ của đối tượng View.
- Id có thể giống nhau cho các điều khiển khác nhau trong cùng một tập tin giao diện.
- Phương thức thiết lập
  - setId
- Phương thức truy xuất
  - getId





## 1.3 Thuộc tính

- ❑ Thuộc tính Id được đi kèm với đối tượng View khi khai báo trong XML cho phép truy xuất trong Java Code khi cần.

- Ví dụ:

- Khai báo id trong XML

```
<Button
```

```
    android:id="@+id/my_button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/my_button_text"/>
```

- Truy xuất trong JavaCode

```
Button myBtn = (Button) findViewById(R.id.my_button);
```



## 1.3 Thuộc tính

- ❑ Vị trí: cho biết toạ độ hiển thị cho View trên giao diện.

- Phương thức thiết lập

- layout
  - setLeft
  - setTop
  - setRight
  - setBottom

- Phương thức truy xuất:

- getLeft
  - getTop
  - getRight
  - getBottom

- ❑ Vị trí của View tuỳ thuộc vào thuộc tính của đối tượng Layout.





## 1.3 Thuộc tính

- ❑ **Kích thước: bao gồm chiều ngang và chiều cao của một đối tượng View.**

- Kích thước của đối tượng View có thể thiết lập qua 3 thông số:
  - WRAP\_CONTENT
  - MATCH\_PARENT (API 8 trở lên)
  - FILL\_PARENT
  - Một con số bất kỳ (tính theo dp/px/dip).



## 1.3 Thuộc tính

- ❑ **Kích thước: bao gồm chiều ngang và chiều cao của một đối tượng View.**

- Phương thức thiết lập trong Java Code:
  - Thiết lập thông qua đối tượng LayoutParams
- Thuộc tính thiết lập trong XML:
  - layout\_width
  - layout\_height
- Phương thức truy xuất:
  - getWidth
  - getHeight
  - getMeasuredWidth
  - getMeasureHeight



## 1.3 Thuộc tính



### □ Canh lề nội dung trong JavaCode:

- Phương thức thiết lập:
  - setPadding
- Phương thức truy xuất:
  - getPaddingTop
  - getPaddingLeft
  - getPaddingRight
  - getPaddingBottom



## Nội dung



### 1. Khái niệm View

### 2. Các điều khiển cơ bản

- TextView
- Button





## 2.1 TextView

❑ Đổi tượng cho phép hiển thị các nội dung văn bản ở 4 dạng:

- Normal
- SmallText
- MediumText
- LargeText

❑ Thiết lập nội dung hiển thị:

- Trong Java code:
  - `textView.setText("Đổi tượng TextView");`
- Trong XML:
  - `android:text="Đổi tượng TextView"`



## 2.1 TextView

❑ Cơ chế tự động thiết lập hành động cho các siêu liên kết, bao gồm:

- Web
- Email
- Phone
- Map

❑ Phương thức thiết lập

- Trong Java code:
  - `textView.setAutoLinkMask(Linkify.PHONE_NUMBERS);`
- Trong XML:
  - `android:autoLink="phone"`



## 2.1 TextView



### ❑ Cho phép hiển thị hình ảnh theo văn bản ở hai dạng:

- Theo bố cục văn bản: Left, Top, Right, Bottom
- Theo đoạn văn bản: Start, End

### ❑ Phương thức thiết lập

- Trong Java code:
  - `textView.setCompoundDrawables(Left, Top, Right, Bottom);`
- Trong XML:
  - `android:drawableLeft="@drawable/ic_launcher"`



## 2.1 TextView



### ❑ Một số phương thức quan trọng:

- `setTextColor – android:textColor`
- `setTextSize – android:textSize`
- `setTypeFace – android:typeface`



## 2.2 Button



- ❑ Đổi tượng Button được xây dựng từ TextView, cho phép thể hiện các nội dung văn bản, hình ảnh – nhận và phản hồi tương tác nhấn từ người dùng:
- ❑ Các dạng Button:
  - Button
  - CompoundButton
    - CheckBox
    - RadioButton
    - ToggleButton
    - Switch



## 2.2 Button



- ❑ Button:
  - Lắng nghe sự kiện nhấn trong Java-Code:

```
button.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Log.i("HTSI", "onClick");  
    }  
});
```
  - Lắng nghe sự kiện nhấn trong XML:  
`android:onClick="tenPhuongThuc"`



## 2.2 Button



### ❑ CompoundButton:

- Checkbox: đối tượng nút bấm hai trạng thái “được chọn” và “bỏ chọn”.
- Phương thức lắng nghe sự kiện thay đổi trạng thái:

```
checkBox.setOnCheckedChangeListener(new OnCheckedChangeListener() {
    @Override
    public void onCheckedChange(CompoundButton v, boolean isChecked) {
        Log.i("HTSI", "onChecked");
    }
});
```



## 2.2 Button



### ❑ CompoundButton:

- RadioButton: đối tượng nút bấm hai trạng thái “được chọn” và “bỏ chọn”, không thể “bỏ chọn” khi đã “được chọn”.
- Thường xử lý trên nhóm nút nhiều trạng thái.
- Phương thức lắng nghe sự kiện thay đổi trạng thái trên nhóm nút:

```
radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener() {
    @Override
    public void onCheckedChange(RadioGroup group, int checkedId) {
        Log.i("HTSI", "Checked at id:" + checkedId);
    }
});
```



## 2.2 Button



### ❑ CompoundButton:

- ToggleButton: đối tượng nút bấm hai trạng thái “bật” và “tắt”, thể hiện trạng thái trên đối tượng.
- Thuộc tính quan trọng:
  - textOn: trạng thái nút đang bật
  - textOff: trạng thái nút đang tắt
- Phương thức lắng nghe sự kiện thay đổi trạng thái trên nhóm nút:

```
toggleButton.setOnCheckedChangeListener(new OnCheckedChangeListener() {
    @Override
    public void onCheckedChange(CompoundButton v, boolean isChecked) {
        Log.i("HTSI", "onChecked");
    }
});
```



## 2.2 Button



### ❑ CompoundButton:

- Switch: đối tượng nút bấm hai trạng thái “bật” và “tắt”, có thể thao tác bằng cách trượt ngón tay trên đối tượng.
- Thuộc tính quan trọng:
  - textOn: trạng thái nút đang bật
  - textOff: trạng thái nút đang tắt
- Phương thức lắng nghe sự kiện thay đổi trạng thái trên nhóm nút:

```
switchButton.setOnCheckedChangeListener(new OnCheckedChangeListener() {
    @Override
    public void onCheckedChange(CompoundButton v, boolean isChecked) {
        Log.i("HTSI", "onChecked");
    }
});
```



## Thảo luận





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

## Lập trình Android

### Bài 4. Tài nguyên ứng dụng cơ bản

*Ngành Mạng & Thiết bị di động*

cuu duong than cong . com

2014



5014



cuu duong than cong . com

## Nội dung



### 1. Khái niệm

- Tài nguyên & Tính tương thích
- Định nghĩa tài nguyên
- Truy xuất tài nguyên
- Tài nguyên Alias

### 2. Các tài nguyên cơ bản



### 1.1 Tài nguyên và tính tương thích



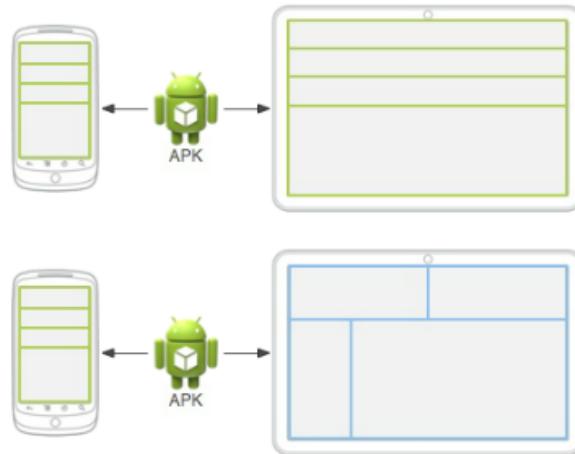
- **Tài nguyên:** một dạng dữ liệu được xây dựng nhằm đáp ứng các yêu cầu về hiển thị bao gồm hình ảnh, âm thanh, văn bản, các bộ cục...tương thích cho từng thiết bị riêng biệt.
- **Cho phép khai báo một lần và sử dụng trong phạm vi toàn ứng dụng,** dễ dàng thay đổi theo ngữ cảnh.
- **Tính tương thích:** để có thể tối ưu hóa tính tương thích thiết bị tài nguyên được chia làm hai dạng:
  - Tài nguyên mặc định: không quan tâm đến cấu hình của thiết bị hoặc không có tài nguyên để lựa chọn.
  - Tài nguyên đặc trưng: được sử dụng trên thiết bị riêng biệt thông qua các từ hạn định và đường dẫn.





## 1.1 Tài nguyên và tính tương thích

- Ứng dụng tự lựa chọn tài nguyên phù hợp với thông tin cấu hình thiết bị, tài nguyên mặc định được chọn nếu không có tài nguyên phù hợp.



## 1.2 Định nghĩa tài nguyên

- Tài nguyên ứng dụng được định nghĩa trong thư mục **res** của dự án, bao gồm các dạng tài nguyên sau:
  - Animator
  - Anim
  - Color
  - Drawable
  - Layout
  - Menu
  - Raw
  - Values
  - XML



## 1.2 Định nghĩa tài nguyên



### ❑ Vấn đề về định nghĩa tài nguyên:

- Có quá nhiều thiết bị có cấu hình khác nhau về kích thước màn hình, độ phân giải, phím vật lý...
- Mỗi thiết bị có thể hoạt động ở nhiều chế độ khác nhau: nằm ngang, nằm đứng, thay đổi ngôn ngữ...

### ❑ Từ hạn định: dùng để tạo ra các tài nguyên khác nhau cho nhiều thiết bị có cấu hình khác nhau hoạt động ở các chế độ khác nhau.

- Ví dụ:

drawable

ic\_launcher.png

drawable-hdpi

ic\_launcher.png



## 1.2 Định nghĩa tài nguyên



### ❑ Các dạng từ hạn định:

- MCC – MNC
  - mcc452-mnc04
- Language & Region
  - vi-rVN
- Layout Direction (API 17)
  - ldltr - ldrtl
- Smallest Width
  - sw320dp – sw480dp – sw600dp - sw720dp
- Available Width
  - w720dp – w1024dp
- Available Height
  - h720dp – h1024dp





## 1.2 Định nghĩa tài nguyên

### ❑ Các dạng từ hạn định:

- Screen Size
  - small – mormal – large - xlarge
- Screen Aspect
  - long - notlong
- Screen Orientation
  - port- land
- UI Mode
  - car – desk – television – appliance
- Night Mode
  - night – notnight
- Screen pixel density
  - ldpi – mdpi – hdpi – xhdpi – nodpi - tvdpi



## 1.2 Định nghĩa tài nguyên

### ❑ Các dạng từ hạn định:

- Touch Screen
  - notouch - finger
- Keyboard
  - keysexposed – keyshidden - keyssoft
- Input Method
  - nokeys – qwerty – 12key
- Navigation Key
  - Navexposed – navhidden
- Non-Touch Navigation
  - nonav – dpad – trackball- wheel
- Platform Version
  - v3 – v4 – v7 – v11...



## 1.2 Định nghĩa tài nguyên



### ❑ Cách tạo tài nguyên:

- Tạo thư mục mới trong thư mục **res** với định dạng:  
`<tên tài nguyên>-<tùy chọn định danh>`
- Ví dụ:  
`drawable-vi-rVN`

### ❑ Qui tắc đặt tên cho thư mục tài nguyên:

- Có thể có nhiều tùy chọn định danh cho một thư mục tài nguyên cách nhau bằng dấu gạch ngang ("").
- Các tùy chọn định danh phải theo thứ tự ưu tiên.
- Các thư mục tài nguyên không được chứa thư mục tài nguyên khác.
- Không cho phép hai tùy chọn định danh giống nhau trên cùng một thư mục.



## 1.3 Truy xuất tài nguyên



### ❑ Tất cả tài nguyên ứng dụng được truy xuất thông qua lớp R.

### ❑ Lớp R:

- Lớp tĩnh.
- Chứa trong thư mục gen, tự động tạo các định danh cho tài nguyên (ID) thông qua AAPT (Android Application Project Tool).
- Chứa các lớp tài nguyên, mỗi dạng tài nguyên là một lớp tĩnh.
  - Ví dụ:
    - Truy xuất tài nguyên hình ảnh:
      - Java code: R.drawable.ic\_launcher
      - XML: @drawable/ic\_launcher



## 1.3 Truy xuất tài nguyên



- ❑ Cú pháp dùng chung khi truy xuất:

Java Code:

```
[<package_name.>]R.<resource_type>.<resource_name>
```

XML:

```
@[<package_name.>]<resource_type>/<resource_name>
```

Trong đó:

- Package\_name: tên gói ứng dụng
- Resource\_type: dạng tài nguyên
- Resource\_name:
  - Tên tài nguyên cần truy xuất không bao gồm phần mở rộng tập tin
  - Thuộc tính **android:name** dành cho các tài nguyên cơ bản (string, color...).



## 1.4 Tài nguyên Alias



- ❑ Cho phép tạo ra tài nguyên từ tài nguyên có sẵn, phục vụ cho nhiều cấu hình thiết bị nhưng không phải là tài nguyên mặc định.

- ❑ Ví dụ:

- Vấn đề: tạo biểu tượng ứng dụng khác nhau cho các ngôn ngữ khác nhau, đối với tiếng Anh và tiếng Việt thì cùng biểu tượng.
  - Giải quyết vấn đề (không dùng Alias):
    - Tạo thư mục tài nguyên cho từng ngôn ngữ.
    - Chép hình ảnh khác nhau cho từng thư mục, hai thư mục có từ hạn định **en** và **vi** có hình ảnh giống nhau.
  - Giải quyết vấn đề (dùng Alias):
    - Tạo thư mục tài nguyên cho từng ngôn ngữ.
    - Chép hình ảnh khác nhau cho từng thư mục.
    - Thư mục có từ hạn định **en** tạo tài nguyên Alias từ thư mục **vi**.



## Nội dung



1. Khái niệm
2. Các tài nguyên cơ bản
  - String
  - Color
  - Dimen
  - Array



## 2. Các tài nguyên cơ bản



- ❑ Các tài nguyên cơ bản được lưu trữ trong thư mục **res/values**.
- ❑ Định danh tài nguyên được khởi tạo thông qua thuộc tính **name**, không phải tên tập tin.
- ❑ Có thể lưu trữ nhiều tài nguyên vào trong một tập tin.
- ❑ Một số tên tập tin đề xuất trong values:
  - string.xml
  - arrays.xml
  - colors.xml
  - dimens.xml
- ❑ Tất cả các tập tin xml trong values, được mở đầu và kết thúc bằng cặp thẻ **<resource></resource>**.



## 2.1 String



- ❑ Cung cấp tài nguyên dạng văn bản cho ứng dụng, cho phép thực hiện các thao tác định dạng và thiết kế khác nhau, bao gồm ba dạng:
  - String
  - StringArray
  - QuantityString (Plural)



## 2.1 String



### ❑ String:

- Khai báo:

```
<string name="string_name">Text_string</string>
```

Trong đó:

- string\_name: định danh dùng để truy xuất trong XML và Java Code.
- text\_string: nội dung lưu trữ.

- Ví dụ: tập tin **strings.xml**

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<resource>
```

```
    <string name="hello">Hello world!</string>
```

```
</resource>
```



## 2.1 String



### □ String:

- Ví dụ truy xuất và sử dụng: Khai báo TextView và gắn văn bản cho thuộc tính text.

- Truy xuất trong XML:

```
<TextView
```

```
...
```

```
    android:text="@string/hello" />
```

- Truy xuất trong Java Code:

Truy xuất trực tiếp:

```
textView.setText(R.string.hello);
```

Dùng phương thức getString:

```
textView.setText(getString(R.string.hello));
```



## 2.1 String



### □ StringArray:

- Khai báo:

```
<string-array name="string_array_name">
```

```
    <item>Text_string</item>
```

```
</string-array>
```

Trong đó:

- string\_array\_name: định danh dùng để truy xuất trong XML và Java Code.
- text\_string: nội dung lưu trữ cho từng item.

- Ví dụ:

```
<string-array name="android_courses">
```

```
    <item>Basic Android</item>
```

```
    <item>Advanced Android</item>
```

```
</string-array>
```



## 2.1 String



### ❑ **StringArray:**

- Truy xuất trong Java Code:

```
Resources res = getResources();  
String[] androidCourses =  
    res.getStringArray(R.array.android_courses);
```



## 2.1 String



### ❑ **Quantity: được sử dụng cùng bộ số đếm tùy thuộc vào qui ước của từng ngôn ngữ, bao gồm các bộ đếm:**

- Zero
- One
- Two
- Few
- Many
- Other

### ❑ **Thường được dùng kết hợp với các định dạng số nguyên.**

### ❑ **Hệ thống tự động lựa chọn tùy chọn bộ đếm tùy thuộc vào số đếm và ngôn ngữ sử dụng trên thiết bị.**



## 2.1 String



### □ Quantity:

- Khai báo:

```
<plurals name="plural_name">
    <item quantity="count">Text_string</item>
</plurals>
```

Trong đó:

- plural\_name: định danh dùng để truy xuất trong XML và Java Code.
- count: bộ đếm sử dụng.
- text\_string: nội dung lưu trữ cho từng item.



## 2.1 String



### □ Quantity:

- Ví dụ:

```
<plurals name="numberOfBooks">
    <item quantity="one">One book found.</item>
    <item quantity="other">%d books found.</item>
</plurals>
```

- Truy xuất trong Java Code:

```
Resources res = getResources();
String booksFound =
    res.getQuantityString(R.plurals.numberOfBooks, 2, 2);
```



## 2.1 String



### ❑ Định dạng String:

- Truyền tham số: cho phép thực hiện tạo các đoạn văn bản có chứa tham số truyền vào.
- Ví dụ:

```
<string name="messages">  
    Chào %1$s! Bạn có %2$d tin nhắn mới.  
</string>
```

- Truy xuất trong Java Code:

```
String messages = getString(R.string.messages);  
messages = String.format(messages, "HTSI", 10);  
Log.d("HTSI", messages);  
// Chào HTSI! Bạn có 10 tin nhắn mới.
```



## 2.1 String



### ❑ Định dạng String:

- Định dạng HTML: cho phép hiển thị các định dạng cấu trúc HTML.
- Ví dụ:

```
<string name="messages">  
    Chào %1$s! Bạn có &lt;b>%2$d tin nhắn mới &lt;/b>.  
</string>
```

- Truy xuất trong Java Code:

```
String messages = getString(R.string.messages);  
messages = String.format(messages, "HTSI", 10);  
CharSequence htmlMessages = Html.fromHtml(messages);  
Log.d("HTSI", htmlMessages);  
// Chào HTSI! Bạn có 10 tin nhắn mới.
```





## 2.2 Color

- ❑ Khai báo tài nguyên sử dụng cho các hiển thị màu sắc như phông nền, hình ảnh, màu chữ...
- ❑ Có thể sử dụng các định dạng màu sắc sau:
  - #RGB
  - #ARGB
  - #RRGGBB
  - #AARRGGBB



## 2.2 Color

- ❑ Khai báo:

```
<color name="color_name">hex_color</color>
```

Trong đó:

- color\_name: định danh dùng để truy xuất trong XML và Java Code.
- hex\_color: định dạng màu sắc

- Ví dụ: tập tin **colors.xml**

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<resource>
```

```
    <color name="my_blue">#33B5E5</color>
```

```
    <color name="my_green">#99CC00</color>
```

```
</resource>
```





## 2.2 Color

- ❑ Ví dụ truy xuất và sử dụng: Khai báo TextView và thiết lập màu văn bản thông qua thuộc tính textColor.

- Truy xuất trong XML:

```
<TextView
```

```
...
```

```
    android:textColor="@color/my_blue" />
```

- Truy xuất trong Java Code:

Dùng phương thức getColor:

```
Resources res = getResources();  
int myBlue = res.getColor(R.color.my_blue);  
textView.setTextColor(myBlue);
```



## 2.2 Color

- ❑ ColorStateList: đối tượng cho phép xây dựng một tập các màu sắc khác nhau hiển thị cho các trạng thái khác nhau:

- Pressed
  - Focused
  - Selected
  - Checkable
  - Checked
  - Enable
  - Window\_focused

- ❑ Khai báo trong thư mục res/color, tập tin XML khai báo trong thư mục này bắt đầu và kết thúc bằng cặp thẻ <selector></selector>.

- ❑ Định danh ColorStateList được truy xuất thông tên tập tin (không bao gồm phần mở rộng).



## 2.2 Color



### ❑ ColorStateList:

- Khai báo:

```
<selector
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item [android:state = "true|false"]
        android:color="hex_color" />
</selector>
```

Trong đó:

- state: chỉ định trạng thái thông qua **true** hoặc **false**, nếu không có chỉ định trạng thái là bình thường.
- hexa\_color: màu tương ứng với trạng thái.



## 2.2 Color



### ❑ ColorStateList:

- Ví dụ: tập tin **button\_text\_selector.xml** trong **res/color**

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true" android:color="@color/my_green"/>
    <item android:color="@color/my_blue"/>
</selector>
```

- Truy xuất sử dụng trong XML

```
<button
    ...
    android:textColor="@color/button_text_selector"
/>
```



## 2.3 Dimen



- ❑ Khai báo tài nguyên sử dụng cho các đại lượng kích thước trong ứng dụng.
- ❑ Có thể sử dụng các đại lượng kích thước sau:
  - dp – dip
  - sp
  - pt
  - Px
  - mm
  - in



## 2.3 Dimen



- ❑ Khai báo:

```
<dimen name="dimen_name">size</dimen>
```

Trong đó:

- dimen\_name: định danh dùng để truy xuất trong XML và Java Code.
- size: đại lượng đi kèm định dạng

- Ví dụ: tập tin **dimens.xml**

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<resource>
```

```
    <dimen name="text_size">20sp</dimen>
```

```
</resource>
```





## 2.3 Dimen

- Ví dụ truy xuất và sử dụng: Khai báo Button và thiết lập kích thước văn bản thông qua thuộc tính textSize.

- Truy xuất trong XML:

```
<TextView
```

```
...
```

```
    android:textSize="@dimen/text_size" />
```

- Truy xuất trong Java Code:

Dùng phương thức getDimension:

```
Resources res = getResources();  
float textSize = res.getDimension(R.dimen.text_size);  
textView.setTextSize(textSize);
```



## 2.4 TypedArray

- Khai báo tài nguyên cho phép xây dựng tập hợp (mảng) lưu trữ các loại tài nguyên như hình ảnh, màu sắc... hoặc có thể lưu trữ cùng lúc nhiều dạng tài nguyên khác nhau.

- Khai báo:

```
<array name="array_name">  
    <item>value</item>  
</array>
```

Trong đó:

- array\_name: định danh dùng để truy xuất trong XML và Java Code.
- value: giá trị lưu trữ



## 2.4 TypedArray



### ❑ Ví dụ: tập tin arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<resource>  
    <array name="colors">  
        <item>@color/my_blue</item>  
        <item>@color/my_green</item>  
        <item>#FFBB33</item>  
    </array>  
</resource>
```

Truy xuất và sử dụng trong Java Code:

```
TypedArray colors = res.obtainTypedArray(R.array.colors);  
int myBlue = colors.getColor(2,0);
```



## Thảo luận





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

## Lập trình Android

### Bài 5. Intent

*Ngành Mạng & Thiết bị di động*

cuu duong than cong . com  
2014



5014



cuu duong than cong . com

## Nội dung



### 1. Khái niệm về Intent

- Cơ chế hoạt động
- Các dạng Intent
- Xây dựng Intent
- Truy xuất Intent
- Gửi và Phản hồi Intent trong Activity

### 2. Intent Filter



### 1.1 Cơ chế hoạt động



- ❑ Intent được sử dụng để truyền tải thông điệp, yêu cầu một hành động xử lý từ thành phần được gọi.
- ❑ Intent được sử dụng trong ba trường hợp chính:
  - Khởi động Activity thông qua phương thức startActivityForResult.
  - Khởi động Service thông qua phương thức startService.
  - Chuyển thông điệp đến BroadcastReceiver thông qua phương thức sendBroadcast.

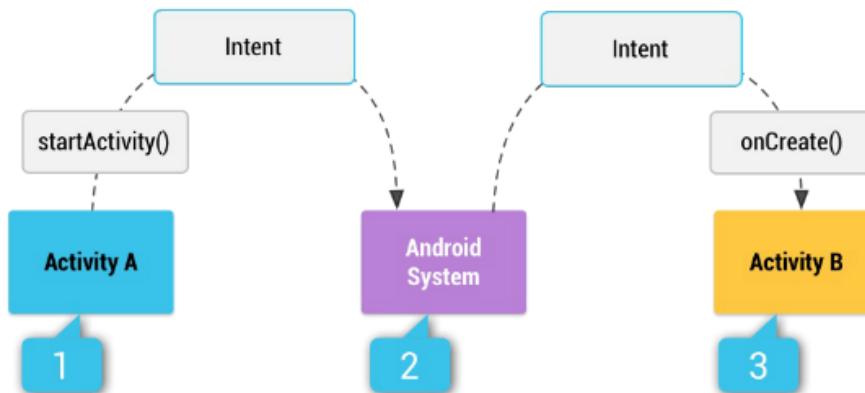




## 1.2 Các dạng Intent

### □ Intent được chia làm hai dạng:

- Explicit Intent: chỉ định rõ thành phần xử lý thông qua tên lớp, thường được dùng để gọi đến các thành phần trong cùng ứng dụng.
- Implicit Intent: không chỉ định rõ thành phần xử lý, thay vào đó bổ sung các thuộc tính như: mô tả hành động, dạng dữ liệu...



## 1.3 Xây dựng Intent

### □ Đối tượng Intent khởi động các thành phần trong ứng dụng đồng thời mang các thông tin về dữ liệu được xử lý, bao gồm các thành phần sau:

- Component: tên thành phần nhận và xử lý Intent
- Action: hành động yêu cầu thực thi
- Data: dữ liệu yêu cầu nhận và xử lý
- Category: mô tả lĩnh vực hoạt động
- Extras: bộ key/value cho phép gửi nhận thông tin
- Flag: biến cờ mô tả cách thức hoạt động





## 1.3 Xây dựng Intent

### ❑ Explicit Intent: chỉ cần sử dụng thuộc tính Component.

- Khai báo:

```
Intent intent = new Intent(this, <Component>);
```

- Ví dụ: khởi động Activity có tên SecondActivity từ MainActivity

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
startActivity(intent);
```



## 1.3 Xây dựng Intent

### ❑ Implicit Intent: chỉ cần sử dụng thuộc tính Action.

- Khai báo:

```
Intent intent = new Intent(<Action>);
```

- Ví dụ: khởi động Activity có thể thực hiện ACTION\_VIEW.

```
Intent intent = new Intent(Intent.ACTION_VIEW);
startActivity(intent);
```





## 1.3 Xây dựng Intent

- Action: một số Action thường dùng trong Intent.

- ACTION\_VIEW
- ACTION\_DIAL
- ACTION\_CALL
- ACTION\_EDIT
- ACTION\_DELETE
- ACTION\_SEND
- ACTION\_SENDTO



## 1.3 Xây dựng Intent

- Data: một dạng đường dẫn URI, cho phép trả đến bảng dữ liệu và truy xuất thông tin bao gồm:

- type
- scheme + authority + path

- Data có thể chỉ định thông qua ba phương thức:

- setData
- setType
- setDataAndType

- Ví dụ: thực hiện cuộc gọi thông qua dữ liệu số điện thoại

```
Intent callPhone = new Intent(Intent.ACTION_CALL);
callPhone.setData(Uri.parse("tel:01234-56789"));
startActivity(callPhone);
```



## 1.3 Xây dựng Intent



- Extras: bao gồm biến Bundle chứa các giá trị bổ sung cần thiết cho thành phần nhận xử lý Intent.
- Có hai cách gửi dữ liệu vào Intent:
  - Trực tiếp:
    - Dùng phương thức putExtra(Key, Value) thiết lập trực tiếp vào Intent.
  - Thông qua Bundle
    - Tạo đối tượng Bundle, dùng phương thức set<KDL>(Key, Value) vào đối tượng Bundle
    - Dùng phương thức putExtras() gửi Bundle vào Intent.



## 1.3 Xây dựng Intent



- Extras:
  - Ví dụ: gửi số nguyên x vào Intent
    - Trực tiếp:
 

```
Intent intent = new Intent();
intent.putExtra("SoNguyenX", x);
```
    - Thông qua Bundle:
 

```
Intent intent = new Intent();
Bundle bundle = new Bundle();
bundle.putInt("SoNguyenX", x);
intent.putExtras(bundle);
```



## 1.4 Truy xuất



### □ Truy xuất:

- Truy xuất dữ liệu trực tiếp Extras:
  - Dùng phương thức `get<KDL>Extra(Key, DefaultValue)` để truy xuất dữ liệu Intent.
- Thông qua Bundle
  - Dùng phương thức `getExtras()` để truy xuất đối tượng Bundle trong Intent.
  - Dùng phương thức `get<KDL>(Key, DefaultValue)` để truy xuất dữ liệu trong Bundle.



## 1.4 Truy xuất



### □ Truy xuất:

- Ví dụ: truy xuất số nguyên được gửi trong Intent

- Trực tiếp:

```
Intent intent = getIntent();
int soNguyenX = intent.getIntExtra("SoNguyenX", 0);


- Thông qua Bundle:


Intent intent = getIntent();
Bundle bundle = intent.getExtras();
int soNguyenX = bundle.getInt("SoNguyenX", 0);
```



## 1.5 Gửi và phản hồi Intent trong Activity



### ❑ Việc gửi và phản hồi Intent trong Activity được chia làm 3 bước

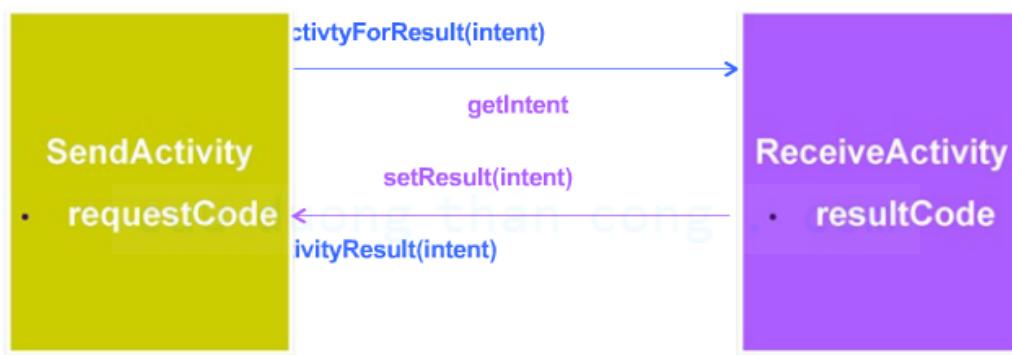
- Bước 1: Gửi Intent thông qua phương thức `startActivityForResult()` bao gồm 2 tham số:
  - Intent: dữ liệu cần gửi để xử lý.
  - requestCode: mã yêu cầu xử lý từ phía gửi.
- Bước 2: Nhận và xử lý Intent, sau đó xác nhận thông tin phản hồi thông qua phương thức `setResult()` trong thành phần ứng dụng phản hồi.
  - Khởi tạo đối tượng Intent, thiết lập các thuộc tính cần thiết: action, category...
  - Gửi dữ liệu phản hồi trực tiếp vào Intent hoặc thông qua biến Bundle.
  - Gọi phương thức setResult với tham số truyền vào là Intent.
- Bước 3: Gọi phương thức `onActivityResult()` truy xuất ba tham số:
  - requestCode: mã yêu cầu giải quyết với intent tương ứng.
  - resultCode: mã kết quả nhận về từ phía phản hồi.
  - Intent: dữ liệu nhận về từ phía phản hồi.



## 1.5 Gửi và phản hồi Intent trong Activity



### ❑ Mô hình hoạt động:



## Nội dung



1. Khái niệm về Intent

### 2. Intent Filter

- Mô tả
- Qui tắc thiết lập
- Xây dựng IntentFilter



## 2.1 Mô tả



- Thực hiện mô tả cấu trúc Intent, cho phép thực hiện chỉ nhận các Intent theo đúng cấu trúc đã mô tả.
- Có thể lọc Intent theo ba thuộc tính:
  - Action
  - Data (type, scheme, authority & path)
  - Category





## 2.2 Qui tắc thiết lập

- ❑ IntentFilter thực hiện lọc Intent theo thứ tự ưu tiên khi có nhiều thuộc tính được thiết lập và có những qui tắc nhất định:
  - Nếu không thiết lập Action, chỉ nhận các Intent không có Action.
  - Nếu thiết lập thuộc tính Action và không thiết lập thuộc tính Data, chỉ cho phép lọc các Intent không có Data.
  - IntentFilter cho phép nhận các Intent có bất kỳ dữ liệu nào có liên quan đến thuộc tính Action.



## 2.3 Xây dựng IntentFilter

- ❑ Có thể khởi tạo đối tượng IntentFilter bằng 2 cách:
  - Trong java Code:
    - Các hàm khởi tạo:
      - IntentFilter()
      - IntentFilter(String Action)
      - IntentFilter(String Action, Uri data)
      - IntentFilter(IntentFilter o)
  - Trong tập tin AndroidManifest.xml:
    - Khai báo thẻ cặp thẻ <intentfilter></intentfilter>
    - Trong cặp thẻ có thể chứa các thẻ sau:
      - <action/>
      - <data/>
      - <category/>



## 2.3 Xây dựng IntentFilter



### □ Action:

- Các thuộc tính:
  - <action android:name="string" />

- Trong đó:
  - android:name : sử dụng các thuộc tính trong lớp Intent.ACTION\_string hoặc tự định nghĩa chuỗi action.

- Ví dụ khai báo:

```
<action android:name="android.intent.action.MAIN" />
```

```
<action android:name="com.htsi.t3h.action.ShowImage" />
```



## 2.3 Xây dựng IntentFilter



### □ Data:

- Các thuộc tính:
  - <data android:scheme="string"  
  android:host="string"  
  android:port="string"  
  android:path="string"  
  android:pathPattern="string"  
  android:pathPrefix="string"  
  android:mimeType="string" />

- Ví dụ khai báo:

```
<data android:scheme="http" android:mimeType="video/mpeg" />
```

```
<data android:mimeType="image/*" />
```

```
<data android:mimeType="*/*" />
```



## 2.3 Xây dựng IntentFilter



### ❑ Category:

- Các thuộc tính:

- <category android:name="string" />

- Trong đó:

- android:name : Intent.CATEGORY\_string. Khai báo theo cấu trúc android.intent.category.string

- Ví dụ khai báo:

```
<category android:name="android.intent.category.DEFAULT" />
```

```
<category android:name="android.intent.category.LAUNCHER" />
```



## Thảo luận





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

## Lập trình Android

### Bài 6. Tài nguyên ứng dụng hình ảnh & giao diện

*Ngành Mạng & Thiết bị di động*

cuu duong than cong . com

2014



2014



cuu duong than cong . com

## Nội dung



1. Tài nguyên hình ảnh
  - Thư mục lưu trữ - Định dạng – Truy xuất
  - Các dạng tài nguyên hình ảnh
2. Tài nguyên giao diện



### 1.1 Thư mục lưu trữ - Định dạng – Truy xuất



- ❑ **Thư mục lưu trữ:**
  - Các tài nguyên hình ảnh được lưu trữ trong thư mục **res/drawable**.
  - Có thể có nhiều thư mục drawable theo từ hạn định khác nhau:
    - Ví dụ: drawable-hdpi, drawable-xhdpi...
- ❑ **Định dạng:**
  - Tài nguyên hình ảnh bao gồm cả định dạng \*.xml và định dạng hình ảnh (.png, .gif, .jpg).
- ❑ **Truy xuất: bao gồm 2 cách thức:**
  - Java: R.drawable.<tên tài nguyên>.
  - XML: @[pakage:]drawable/<tên tài nguyên>.



## 1.1 Thư mục lưu trữ - Định dạng - Truy xuất



### ❑ Ví dụ truy xuất tài nguyên hình ảnh:

- Java:

```
Resources res = getResources();
Drawable drawable = res.getDrawable(R.drawable.ic_launcher);
```

- XML:

<pre>&lt;ImageView     android:layout_width="50dp"     android:layout_height="50dp"     android:src=         "@drawable/ic_launcher" /&gt;</pre>	<pre>&lt;ImageButton     android:layout_width="50dp"     android:layout_height="50dp"     android:background=         "@drawable/ic_launcher" /&gt;</pre>
--	---



## 1.2 Các dạng tài nguyên hình ảnh



### ❑ Bao gồm các định dạng:

- Bitmap
- Shape
- LayerList
- StateList
- LevelList
- Transition
- Inset
- Clip
- Scale
- Nine-Patch





## 1.2 Các dạng tài nguyên hình ảnh

### ❑ Bitmap:

- Định dạng ảnh nhị phân, Android hỗ trợ ba định dạng tài nguyên hình ảnh: png, jpg và gif.
- Các thực thi của Bitmap bao gồm:
  - Sử dụng như tài nguyên thông qua R.drawable.filename
  - Tham chiếu biến dịch tài nguyên thông qua đối tượng BitmapDrawable.



## 1.2 Các dạng tài nguyên hình ảnh

### ❑ Bitmap:

- Sử dụng các thuộc tính Bitmap trong XML:
  - Antialias (XML)
  - Dither
  - Filter
  - Gravity
  - Mipmap
  - Tilemode
  - Automirrored



## 1.2 Các dạng tài nguyên hình ảnh



### ❑ Bitmap:

- Ví dụ xây dựng Bitmap trong XML: [mipmap.xml](#)

```
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
    android:mipMap="false"
    android:src="@drawable/caro"
    android:tileMode="repeat" >
</bitmap>
```

- Truy xuất trong Java code:

```
BitmapDrawable drawable = (BitmapDrawable)getResources()
    .getDrawable(R.drawable.mipmap);
```



## 1.2 Các dạng tài nguyên hình ảnh



### ❑ Shape:

- Tài nguyên hình ảnh cho các đối tượng đa giác được vẽ bằng XML, bao gồm:
  - Rectangle
  - Oval
  - Line
  - Ring
- Tham chiếu biến dịch tài nguyên thông qua đối tượng GradientDrawable.





## 1.2 Các dạng tài nguyên hình ảnh

### □ Shape:

- Sử dụng các thuộc tính để cấu tạo đối tượng:

- Corners (Rectangle) - Integer
  - radius
  - topLeftRadius
  - topRightRadius
  - bottomLeftRadius
  - bottomRadius
- Padding (Rectangle) – Integer
  - left
  - top
  - right
  - bottom



## 1.2 Các dạng tài nguyên hình ảnh

### □ Shape:

- Sử dụng các thuộc tính để cấu tạo đối tượng:

- Gradient
  - angle - integer
  - centerX - integer
  - centerY - integer
  - centerColor - integer
  - endColor - color
  - gradientRadius – integer
  - startColor – color
  - type – linear | radial | sweep
  - useLevel – true | false





## 1.2 Các dạng tài nguyên hình ảnh

### □ Shape:

- Sử dụng các thuộc tính để cấu tạo đối tượng:
  - Size – integer
    - width – integer
    - height – integer
  - Solid – integer
    - color – color
  - Stroke – integer
    - width - integer
    - color – color
    - dashWith – integer
    - dashGap - integer



## 1.2 Các dạng tài nguyên hình ảnh

### □ Shape:

- Một số thuộc tính chỉ sử dụng cho đối tượng Ring:
  - innerRadius
  - innerRadiusRatio
  - thickness
  - thicknessRatio
  - useLevel (false)



## 1.2 Các dạng tài nguyên hình ảnh



### □ Shape:

- Ví dụ:

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >
    <gradient
        android:angle="90"
        android:startColor="@android:color/holo_blue_bright"
        android:type="linear" />
    <corners android:radius="20dp"/>
    <size
        android:height="80dp"
        android:width="80dp" />
</shape>
```



## 1.2 Các dạng tài nguyên hình ảnh



### □ LayerList:

- Tài nguyên hình ảnh cho phép quản lý mảng các đối tượng hình ảnh được vẽ chồng lên nhau, mỗi đối tượng hình ảnh được qui ước là một item.

- Mỗi item bao gồm:
  - drawable – resource
  - Id – resource id
  - top - integer
  - right - integer
  - bottom - integer
  - left - integer

- Tham chiếu biến dịch tài nguyên thông qua đối tượng LayerDrawable.



## 1.2 Các dạng tài nguyên hình ảnh



### □ State List:

- Tài nguyên hình ảnh cho phép quản lý mảng các đối tượng hình ảnh được vẽ theo trạng thái của đối tượng thể hiện.
- Một item bao gồm:
  - drawable – resource
  - Tập các trạng thái có thể có:
    - Pressed
    - Focused
    - Hovered
    - Selected
    - Checkable
    - Enable
    - Activated
    - Window focused
- Tham chiếu biến dịch tài nguyên thông qua đối tượng StateListDrawable.



## 1.2 Các dạng tài nguyên hình ảnh



### □ LevelList:

- Tài nguyên hình ảnh cho phép quản lý mảng các đối tượng hình ảnh, mỗi đối tượng hình ảnh được qui ước là một item, hiển thị ảnh theo cấp độ tương ứng đã khai báo.
- Một item bao gồm:
  - drawable – resource
  - maxLevel
  - minLevel
- Tham chiếu biến dịch tài nguyên thông qua đối tượng LevelListDrawable.





## 1.2 Các dạng tài nguyên hình ảnh

### ❑ Transition:

- Tài nguyên hình ảnh cho phép thực hiện chuyển đổi (hiệu ứng “biến bóng”) giữa hai đối tượng hình ảnh.
- Mỗi item bao gồm:
  - drawable – resource
  - Id – resource id
  - top - integer
  - right - integer
  - bottom - integer
  - left – integer
- Các phương thức xử lý chính:
  - startTransition
  - reverserTransition
  - resetTransition.



## 1.2 Các dạng tài nguyên hình ảnh

### ❑ Inset:

- Tài nguyên hình ảnh cho phép thực hiện lồng đối tượng hình ảnh theo một ví trí cho trước.
- Các thuộc tính bao gồm:
  - drawable – resource
  - insetTop - integer
  - insetRight - integer
  - insetBottom - integer
  - insetLeft – integer
- Tham chiếu biến dịch tài nguyên thông qua đối tượng InsetDrawable.

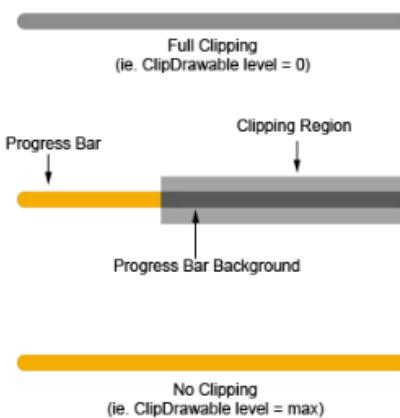


## 1.2 Các dạng tài nguyên hình ảnh



### □ Clip:

- Tài nguyên hình ảnh cho phép thực hiện cắt một đối tượng hình ảnh theo thông số vị trí cho trước, có thể thay đổi thông số cắt trong quá trình hoạt động.
- Tham chiếu biến dịch tài nguyên thông qua đối tượng ClipDrawable.
- Các thuộc tính bao gồm:
  - drawable – resource
  - clipOrientation – integer
  - Gravity
- Các phương thức xử lý chính:
  - setLevel (min:0 – max: 10.000)
  - getLevel

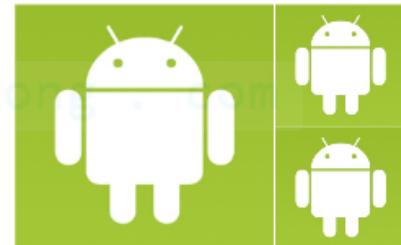


## 1.2 Các dạng tài nguyên hình ảnh



### □ Scale:

- Tài nguyên hình ảnh cho phép thực hiện phóng to hoặc thu nhỏ một đối tượng hình ảnh theo thông số tỉ lệ cho trước, có thể thay đổi thông số tỉ lệ trong quá trình hoạt động.
- Các thuộc tính bao gồm:
  - drawable – resource
  - scaleGravity – integer
  - scaleWidth - %
  - scaleHeight - %
- Tham chiếu biến dịch tài nguyên thông qua đối tượng ScaleDrawable.

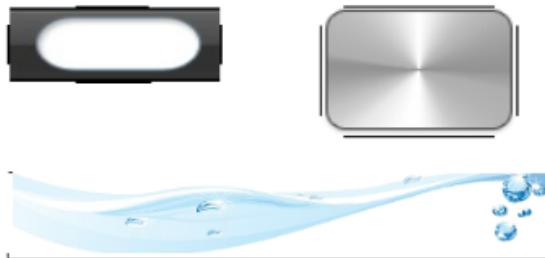


## 1.2 Các dạng tài nguyên hình ảnh



### ❑ NinePatch:

- Tài nguyên hình ảnh cho phép thực hiện tạo đối tượng hình ảnh (PNG) có kích thước co dãn theo tỉ lệ đối tượng thể hiện.
- Các thuộc tính bao gồm:
  - src– resource
  - dither– integer
- Tham chiếu biên dịch tài nguyên thông qua đối tượng NinePatchDrawable.



## Nội dung



### 1. Tài nguyên hình ảnh

### 2. Tài nguyên giao diện

- Thư mục lưu trữ – Truy xuất
- Các định dạng Layout



## 2.1 Thư mục lưu trữ – Truy xuất



### ❑ Thư mục lưu trữ:

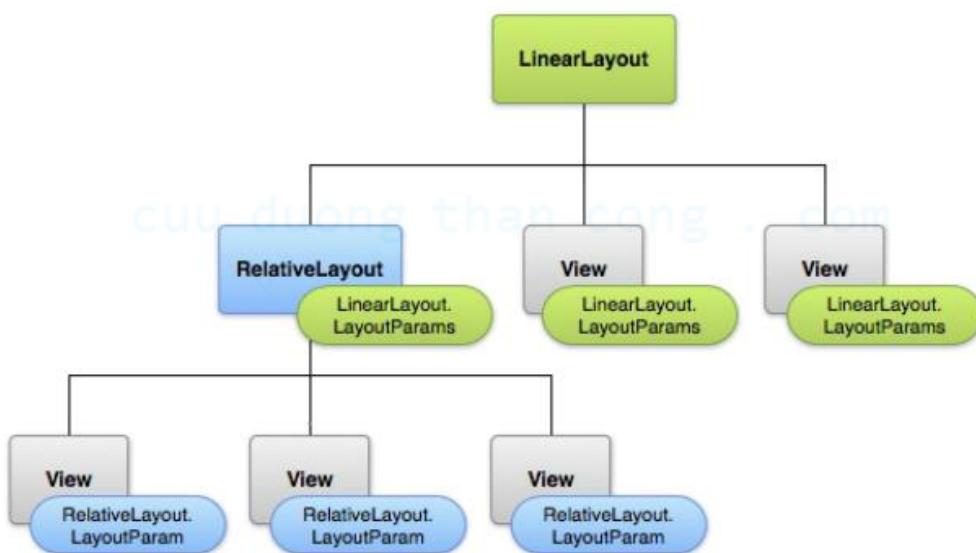
- Các tài nguyên giao diện được lưu trữ trong thư mục **res/layout**.
- Có thể có nhiều thư mục layout theo từ hạn định khác nhau:
  - Ví dụ: layout-land, layout-xhdpi...

### ❑ Truy xuất: bao gồm 2 cách thức:

- Java: R.layout.<tên tài nguyên>.
- XML: @[package:]layout/<tên tài nguyên>.



## 2.1 Thư mục lưu trữ – Truy xuất



## 2.1 Thư mục lưu trữ – Truy xuất



### ❑ Ví dụ khai báo trong XML: activity\_main.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/imgScale"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="@null" />

</LinearLayout>
```



## 2.2 Các định dạng Layout



### ❑ Bao gồm các lớp kế thừa từ ViewGroup:

- AbsoluteLayout (**Deprecated**)
- AdapterView (ListView, GridView...)
- DrawerLayout
- FragmentBreadCrumbs
- **FrameLayout**
- GridLayout
- **LinearLayout**
- PagerTitleStrip
- **RelativeLayout**
- SlidingDrawer
- SlidingPaneLayout
- SwipeRefreshLayout
- ViewPager





## 2.2 Các định dạng Layout

### ❑ FrameLayout:

- Sử dụng trong các trường hợp xây dựng bố cục tổ chức hiển thị một đối tượng duy nhất.
- Đối tượng mặc định vị trí top-left trên FrameLayout, có thể sử dụng thuộc tính Gravity để thiết lập lại vị trí.
- Ví dụ khai báo:

```
<FrameLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" >  
</FrameLayout>
```



## 2.2 Các định dạng Layout

### ❑ FrameLayout:

- Các đối tượng kế thừa phỗ biến:

- **ViewFlipper**: đối tượng cho phép thực hiện hiển thị các đối tượng ở chế độ phân trang, chỉ hiển thị một đối tượng ở một thời điểm.

- Ví dụ khai báo:

```
<ViewFlipper  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" >  
</ViewFlipper>
```

- Các phương thức sử dụng:

- startFlipping
  - setAutoStart
  - showNext
  - showPrevious



## 2.2 Các định dạng Layout



### ❑ FrameLayout:

- Các đối tượng kế thừa phỗ biến:
  - **ScrollView**: đối tượng cho phép thực hiện hiển thị các đối tượng ở chế độ cuộn màn hình, chỉ cho phép chứa một đối tượng ở một thời điểm.
  - Ví dụ khai báo:
 

```
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
</ScrollView>
```
  - Các phương thức sử dụng:
    - setFillViewPort
    - scrollBy
    - scrollTo
    - smoothScrollBy
    - smoothScrollTo



## 2.2 Các định dạng Layout



### ❑ LinearLayout:

- Sử dụng trong các trường hợp xây dựng bố cục tổ chức hiển thị các đối tượng theo một chiều duy nhất (chiều dọc hoặc ngang).
- Đối tượng mặc định vị trí top left trên LinearLayout , có thể sử dụng thuộc tính Gravity để thiết lập lại vị trí.
- Ví dụ khai báo:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
</LinearLayout>
```



## 2.2 Các định dạng Layout



### ❑ LinearLayout:

- **TableLayout:** đối tượng layout kế thừa từ LinearLayout, cho phép hiển thị các đối tượng theo nhiều dòng (TableRow).
- Mỗi dòng có thể chứa nhiều View, mỗi View được xem là một cột.
- Ví dụ khai báo:

```
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <Tablerow>
        <Button/>
    </Tablerow>
</TableLayout>
```



## 2.2 Các định dạng Layout



### ❑ RelativeLayout:

- Sử dụng trong các trường hợp xây dựng bố cục tổ chức hiển thị các đối tượng theo mối quan hệ vị trí.
- Đối tượng được đặt vào RelativeLayout đầu tiên sẽ xác định vị trí cho các đối tượng sau đó.
- Ví dụ khai báo:

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</RelativeLayout>
```



## Thảo luận





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
TRUNG TÂM TIN HỌC

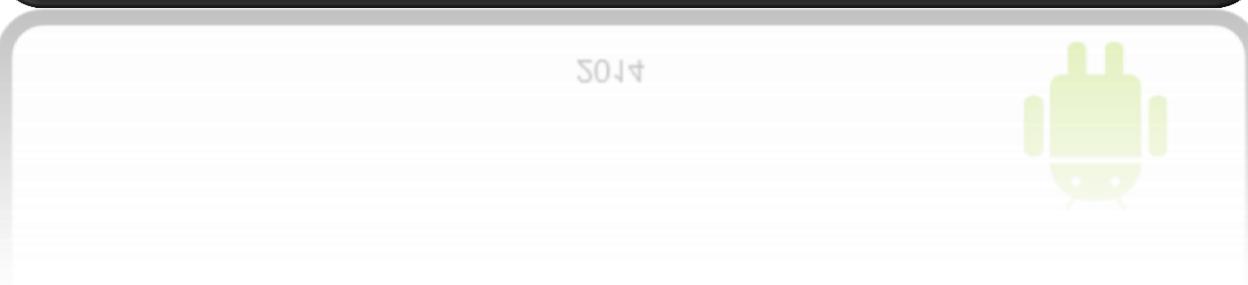
[Go Screen Capture](#)

## Lập trình Android

### Bài 7. Asset – SharedPreference – Bộ nhớ thiết bị

*Ngành Mạng & Thiết bị di động*

cuu duong than cong . com 2014 



cuu duong than cong . com

## Nội dung



### 1. SharedPreference

- Khái niệm lưu trữ
- Khai báo và sử dụng

### 2. Bộ nhớ thiết bị

### 3. Asset



### 1.1 Khái niệm lưu trữ



- ❑ SharedPreference cho phép thực hiện lưu trữ và truy xuất dữ liệu theo cặp khóa key-value cho các kiểu dữ liệu cơ bản.
- ❑ Có thể lưu trữ các kiểu dữ liệu cơ bản sau:
  - Boolean
  - Float
  - Int
  - Long
  - String
- ❑ Dữ liệu vẫn được bảo toàn ngay cả khi ứng dụng bị đóng hoàn toàn.



## 1.2 Khai báo và sử dụng



### ❑ Khai báo truy xuất:

- Có thể sử dụng một trong 2 phương thức sau
  - **getSharedPreferences**: sử dụng truy xuất đến tập tin Preferences đã lưu trữ bằng cách truyền vào tên tập tin.
  - **getPreferences**: truy xuất đến tập tin Preference mặc định tương ứng với Activity gọi thực thi.

### ❑ Mặc định tập tin Preference được lưu trữ theo đường dẫn data/data/<package\_name\_app>/shared\_prefs/<file\_name>.xml



## 1.2 Khai báo và sử dụng



### ❑ Khai báo truy xuất:

- Các định dạng mở tập tin khi truy xuất Preference:
  - MODE\_PRIVATE
  - MODE\_APPEND
  - MODE\_WORLD\_READABLE (Deprecated in API 17)
  - MODE\_WORLD\_WRITEABLE (Deprecated in API 17)
  - MODE\_MULTI\_PROCESS





## 1.2 Khai báo và sử dụng

### ❑ Lưu trữ dữ liệu:

- Tạo đối tượng Editor từ phương thứ **edit()** của Preference
- Lưu trữ dữ liệu vào Preference thông qua phương thức **put<KDL>**:
  - **putBoolean(Key, Value)**
  - **putFloat(Key, Value)**
  - **putInt(Key, Value)**
  - **putLong(Key, Value)**
  - **putString(Key, Value)**
- Ví dụ:

```
boolean wifi_status = checkBoxWifi.isChecked();
SharedPreferences shared_pref = getPreferences(MODE_PRIVATE);
Editor editor = shared_pref.edit();
editor.putBoolean("Wifi_Status", wifi_status);
editor.apply();
```



## 1.2 Khai báo và sử dụng

### ❑ Khai báo truy xuất:

- Các giá trị lưu trữ được truy xuất thông qua phương thức **get<KDL>**:
  - **getBoolean(Key, DefValue)**
  - **getFloat(Key, DefValue)**
  - **getInt(Key, DefValue)**
  - **getLong(Key, DefValue)**
  - **getString(Key, DefValue)**

- Ví dụ:

```
boolean wifi_status =
    getPreferences(MODE_PRIVATE).getBoolean("Wifi_Status");
checkBoxWifi.setChecked(wifi_status);
```



## Nội dung



1. SharedPreference

2. Bộ nhớ thiết bị

- Bộ nhớ trong
- Bộ nhớ ngoài

3. Asset



### 2.1 Bộ nhớ trong



- ❑ **Bộ nhớ trong được cấp phát khi ứng dụng được cài đặt trên thiết bị trên một thư mục riêng biệt:**
- ❑ **Chỉ có ứng dụng mới có thể truy xuất được bộ nhớ của ứng dụng đó.**
  - Đường dẫn các tập tin được lưu trữ:  
`data/data/<package_name_app>/files`
  - Phương thức truy xuất:  
`getFilesDir - File`



## 2.1 Bộ nhớ trong



### ❑ Truy xuất ghi tập tin:

- Gọi phương thức `openFileOutput()` nhận về dữ liệu stream cho đối tượng `FileOutputStream`.
- Đọc dữ liệu với phương thức `read()`.
- Đóng stream bằng phương thức `close()`.

### ❑ Ví dụ:

```
String hello = "Hello Android!";
FileOutputStream fos = openFileOutput("Hello_file.txt",
                                         MODE_APPEND);
fos.write(hello.getBytes());
fos.close();
```



## 2.1 Bộ nhớ trong



### ❑ Truy xuất đọc tập tin:

- Gọi phương thức `openFileInput()` nhận về dữ liệu stream cho đối tượng `FileInputStream`.
- Đọc dữ liệu với phương thức `read()`.
- Đóng stream bằng phương thức `close()`.

### ❑ Ví dụ:

```
FileInputStream fis = openFileInput("Hello_file.txt");
int b = 0; String s = "";
while ((b = fis.read()) != 0)
    s += (char)b;
fis.close();
Log.d("HTSI", s);
```



## 2.2 Bộ nhớ ngoài



- ❑ Mỗi thiết bị Android cung cấp bộ nhớ ngoài cho phép người dùng có thể lưu trữ và truy xuất trực tiếp trên thiết bị hoặc thông qua máy tính khi kết nối USB Storage.
- ❑ Bộ lưu trữ ngoài bao gồm hai dạng:
  - Bộ nhớ thiết bị (non-removable)
  - Bộ nhớ ngoài (sd-card, usb...)
- ❑ Cần xin cấp quyền để truy xuất bộ nhớ này  
`android.permission.WRITE_EXTERNAL_STORAGE`



## 2.2 Bộ nhớ ngoài



- ❑ Ứng dụng được cấp phát bộ nhớ ngoài để lưu trữ dữ liệu, mặc định không thể truy xuất như tập tin Media.
  - Địa chỉ lưu trữ: `/mnt/sdcard/Android/<package_name>/files`
  - Phương thức truy xuất:
    - `getExternalFilesDir(String)` – File
    - `getExternalFilesDirs(String)` – File[] (API 19 - KITKAT)
    - `ContextCompat.getExternalFilesDirs(String)` – File[] (Support Library)
- ❑ Cấu hình cài đặt ứng dụng trên bộ nhớ ngoài thông qua cài đặt `<manifest>` trong `AndroidManifest`.
  - `installLocation`
    - `internalOnly`
    - `Auto`
    - `preferExternal`





## 2.2 Bộ nhớ ngoài

### ❑ Kiểm tra tình trạng bộ nhớ:

- Cần kiểm tra tình trạng bộ nhớ ngoài trước khi thực hiện các thao tác.
- Truy xuất trạng thái bộ nhớ ngoài thông qua phương thức `getExternalStorageState` trong đối tượng `Environment`.
- Các thông số được trả về bao gồm:
  - `MEDIA_UNKNOWN`
  - `MEDIA_REMOVED`
  - `MEDIA_UNMOUNTED`
  - `MEDIA_CHECKING`
  - `MEDIA_NOFS`
  - `MEDIA_MOUNTED`
  - `MEDIA_MOUNTED_READ_ONLY`
  - `MEDIA_SHARED`
  - `MEDIA_BAD_REMOVAL`
  - `MEDIA_UNMOUNTABLE`



## 2.2 Bộ nhớ ngoài

### ❑ Kiểm tra tình trạng bộ nhớ:

- Ví dụ kiểm tra có thẻ đọc ghi trên bộ nhớ ngoài

```
public boolean isExternalStorageWritable() {  
    String state = Environment.getExternalStorageState();  
    if( Environment.MEDIA_MOUNTED.equals(state))  
        return true;  
    return false;  
}
```





## 2.2 Bộ nhớ ngoài

### ❑ Một số phương thức hỗ trợ duyệt tập tin trong bộ nhớ:

- isFile() – boolean
- isDirectory() – boolean
- getAbsolutePath() - String
- getPath() - String
- list() – String[]
- mkdir() - boolean
- mkdirs() – boolean
- delete() - boolean



## 2.2 Bộ nhớ ngoài

### ❑ Một số thư mục dùng chung:

- Thư mục:
  - DIRECTORY\_ALARMS
  - DIRECTORY\_DCIM
  - DIRECTORY\_DOCUMENTS (API 19)
  - DIRECTORY\_DOWNLOADS
  - DIRECTORY\_MOVIES
  - DIRECTORY\_MUSIC
  - DIRECTORY\_NOTIFICATIONS
  - DIRECTORY\_PICTURES
  - DIRECTORY\_POSTCASTS
  - DIRECTORY\_RINGTONES



## Nội dung



1. SharedPreference
2. Bộ nhớ thiết bị
3. Asset



## 3. Asset



- ❑ Thư mục được sử dụng chứa các tập tin không biên dịch của Project:
  - Truy xuất đối tượng quản lý:
    - `getAssets()` – `AssetManager`
  - Các phương thức xử lý chính
    - `list(String path)` – `String[]`
    - `open(String fileName)` – `InputStream`
    - `open(String fileName, int mode)` – `InputStream`
    - `close()`
  - Các chế độ mở Asset
    - `ACCESS_BUFFER`
    - `ACCESS_RANDOM`
    - `ACCESS_STREAMING`
    - `ACCESS_UNKNOWN`



### 3. Asset



#### ❑ Truy xuất phông chữ thông qua thư mục Asset

- Sử dụng phương thức `createFromAsset`
- Ví dụ:

```
Typeface arial = Typeface.createFromAsset(getAssets(), "font/arial.ttf");  
textView.setTypeface(arial);
```



### Thảo luận





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

## Lập trình Android

### Bài 8. Adapter & AdapterView

*Ngành Mạng & Thiết bị di động*

cuu duong than cong . com

2014



5014



cuu duong than cong . com

## Nội dung



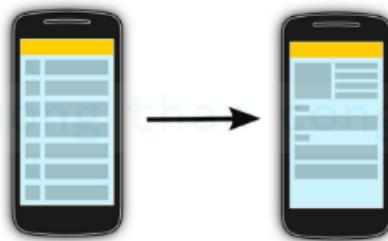
1. Adapter & AdapterView
  - Khái niệm cơ bản
  - Các dạng Adapter
2. AutoCompleteTextView – MultiAutoCompleteTextView
3. Sử dụng Adapter trong AdapterView



### 1.1 Khái niệm cơ bản



- ❑ **AdapterView:** đối tượng điều khiển dạng tập hợp, cho phép hiển thị thông tin cơ bản theo dạng danh sách và thực hiện quản lý thông tin theo từng mục riêng biệt.



- ❑ **Các phương thức quan trọng trên AdapterView:**

- setOnItemClickListener
- setOnItemSelectedListener



## 1.1 Khái niệm cơ bản



- ❑ **Adapter:** đối tượng cơ sở cho phép gắn kết các dữ liệu bên dưới lên các đối tượng điều khiển dạng danh sách (AdapterView).



- ❑ Adapter cho phép thực hiện quản lý giao diện, số lượng chỉ mục trên AdapterView và thực hiện truy vấn dữ liệu, sắp xếp dữ liệu.



## 1.2 Các dạng Adapter



- ❑ Bao gồm các lớp thực thi giao thức Adapter:
  - ArrayAdapter
  - BaseAdapter
  - CursorAdapter
  - HeaderViewListAdapter
  - ResourceCursorAdapter
  - SimpleAdapter
  - SimpleCursorAdapter
  - SpinnerAdapter





## 1.2 Các dạng Adapter

### ❑ Các phương thức xử lý quan trọng trên Adapter:

- getCount - int
- getItems(int position) - Objects
- getItemId(int position) – long
- getView(int position, View convertView, ViewGroup parent) – View



## Nội dung

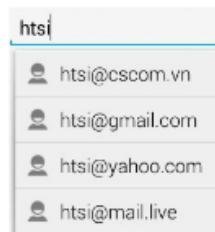
1. Adapter & AdapterView
2. AutoCompleteTextView – MultiAutoCompleteTextView
3. Sử dụng Adapter trong AdapterView



## 2. AutoCompleteTextView



- Đổi tượng kế thừa từ EditText.



- Cho phép xây dựng dữ liệu mẫu hỗ trợ người dùng hoàn chỉnh quá trình nhập liệu trên EditText.
- Thực hiện xây dựng AutoCompleteTextView:
  - Khai báo dữ liệu mẫu.
  - Khai báo giao diện hiển thị cho dữ liệu.
  - Xây dựng Adapter thông qua phương thức khởi tạo tương ứng với dữ liệu và giao diện hiển thị.
  - Thiết lập Adapter cho đối tượng AutoCompleteTextView.



## 2. AutoCompleteTextView



- Ví dụ xây dựng AutoCompleteTextView:

```
// Khởi tạo dữ liệu mẫu
private static final String[] COUNTRIES = new String[]
{"VietNam", "Belgium", "France", "Italy", "Germany", "Spain"};
// Xây dựng Adapter thông qua dữ liệu mẫu và giao diện mẫu
ArrayAdapter<String> adapter = new ArrayAdapter<String> (this,
    android.R.layout.simple_dropdown_item1line, COUNTRIES);
// Tham chiếu điều khiển
AutoCompleteTextView editCountry = (AutoCompleteTextView)
    findViewById(R.id.editCountry);
// Thiết lập Adapter cho điều khiển
editCountry.setAdapter(adapter);
```



## 2. MultiAutoCompleteTextView



- ❑ Đổi tượng kế thừa từ đối tượng AutoCompleteTextView.
- ❑ Cho phép xây dựng dữ liệu mẫu hỗ trợ người dùng hoàn chỉnh quá trình nhập liệu trên EditText.
- ❑ Dữ liệu được hỗ trợ hoàn chỉnh nhiều lần, cách nhau bằng một Tokenizer.
- ❑ Thực hiện xây dựng MultiAutoCompleteTextView:
  - Khai báo dữ liệu mẫu.
  - Khai báo giao diện hiển thị cho dữ liệu.
  - Xây dựng Adapter thông qua phương thức khởi tạo tương ứng với dữ liệu và giao diện hiển thị.
  - Thiết lập Adapter cho đối tượng MultiAutoCompleteTextView.
  - Thiết lập đối tượng Tokenizer.



## 2. MultiAutoCompleteTextView



- ❑ Ví dụ xây dựng MultiAutoCompleteTextView:
- ```

// Khởi tạo dữ liệu mẫu
private static final String[] COUNTRIES = new String[]
{"VietNam", "Belgium", "France", "Italy", "Germny", "Spain"};
// Xây dựng Adapter thông qua dữ liệu mẫu và giao diện mẫu
ArrayAdapter<String> adapter = new ArrayAdapter<String> (this,
    android.R.layout.simple_dropdown_item1line, COUNTRIES);
// Tham chiếu điều khiển
MultiAutoCompleteTextView editCountry = (MultiAutoCompleteTextView)
    findViewById(R.id.editCountry);
// Thiết lập Adapter cho điều khiển
editCountry.setAdapter(adapter);
// Thiết lập Tokenizer
editCountry.setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());

```



## Nội dung



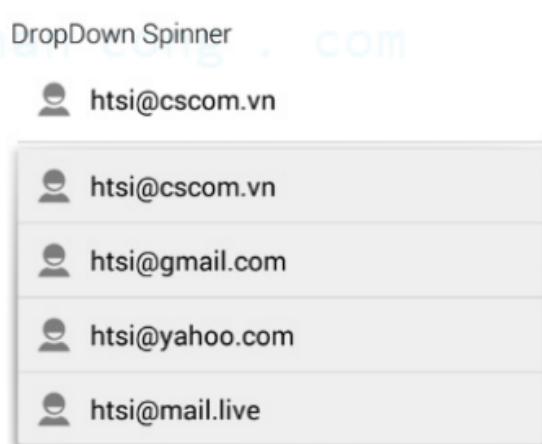
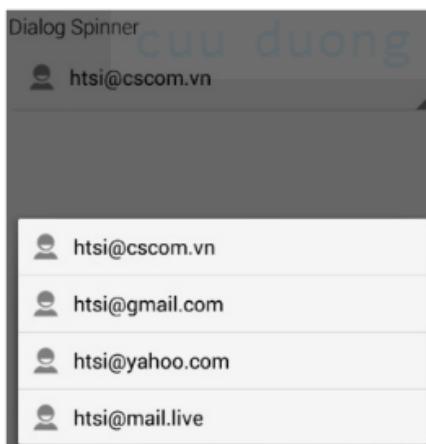
1. Adapter & AdapterView
2. AutoCompleteTextView – MultiAutoCompleteTextView
3. Sử dụng Adapter trong AdapterView



## 3. Sử dụng Adapter trong AdapterView



- ❑ Spinner: đối tượng điều khiển hiển thị một danh mục ở một thời điểm, người dùng có thể lựa chọn một trong nhiều danh mục để hiển thị.
- ❑ Bao gồm hai chế độ hiển thị pop-up lựa chọn (spinnerMode):



### 3. Sử dụng Adapter trong AdapterView



#### ❑ Spinner:

- Thuộc tính XML quan trọng:
  - spinnerMode: dialog | dropdown
  - prompt: string
  - popupBackground: drawable | color
  - gravity
  - entries: string-array
- Một số phương thức quan trọng:
  - setAdapter(SpinnerAdapter)
  - setPrompt(CharSequence) – setPrompt(int resId) (Dialog Mode)
  - setPopupBackgroundResource(int)
  - setPopupBackgroundDrawable(Drawable)



### 3. Sử dụng Adapter trong AdapterView



#### ❑ Spinner:

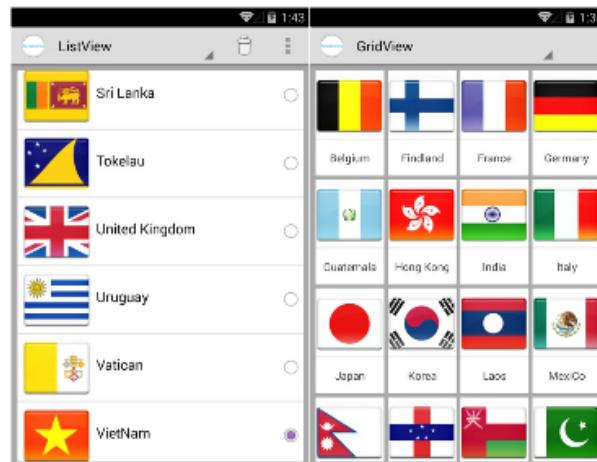
- Ví dụ xây dựng Spinner:

```
// Xây dựng Adapter thông qua dữ liệu tài nguyên và giao diện mẫu
SpinnerAdapter adapter = new ArrayAdapter.createFromResource(this,
    R.array.countries, android.R.layout.simple_dropdown_item_1line);
// Tham chiếu điều khiển
Spinner spinner = (Spinner)findViewById(R.id.spinner);
// Thiết lập Adapter cho điều khiển
spinner.setAdapter(adapter);
```



### 3. Sử dụng Adapter trong AdapterView

- ❑ AbsListView: đối tượng điều khiển hiển thị danh sách các danh mục với thông tin cơ bản, cho phép thực hiện các thao tác khác nhau trên từng danh mục.
- ❑ Bao gồm hai chế độ hiển thị:
  - ListView
  - GridView



### 3. Sử dụng Adapter trong AdapterView

- ❑ AbsListView:
  - Thuộc tính XML quan trọng:
    - listSelector: drawable
    - choiceMode: none | singleChoice | multipleChoice | multipleChoiceModal
    - smoothScrollBar: boolean
    - fastScrollEnable: boolean
  - Một số Interface đã được khai báo sử dụng :
    - TextWatcher
    - ViewTreeObserver.OnGlobalLayoutListener
    - ViewTreeObserver.OnTouchModeChangeListener
    - Filter.FilterListener



### 3. Sử dụng Adapter trong AdapterView



#### ❑ ListView:

- Thuộc tính XML quan trọng:
  - listSelector: drawable
  - divider: drawable
  - dividerHeight: dimen
  - entries: string-array
- Một số phương thức quan trọng:
  - setAdapter(Class Extends <T implements Adapter>)
  - addHeaderView(View) – removeHeaderView(View)
  - addFooterView(View) – removeFooterView(View)
  - setSelection(int)
  - smoothScrollToPosition(int)



### 3. Sử dụng Adapter trong AdapterView



#### ❑ GridView:

- Thuộc tính XML quan trọng:
  - columnWidth: dimen
  - gravity: Gravity
  - horizontalSpacing: dimen
  - verticalSpacing: dimen
  - numColumns: integer
  - stretchMode: none | spacingWidth | columnWidth | spacingWidthUniform
- Một số phương thức quan trọng:
  - setColumnWidth(int) – getColumnWidth()
  - setNumColumn(int) – getNumColumn()
  - setSelection(int)
  - smoothScrollToPosition(int)



## Thảo luận





Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
TRUNG TÂM TIN HỌC

[Go Screen Capture](#)

## Lập trình Android

### Bài 9. *Fragment*

*Ngành Mạng & Thiết bị di động*

cuu duong than cong . com

2014



5014



cuu duong than cong . com

## Nội dung



1. Các khái niệm cơ bản
  - Fragment & Phiên bản hỗ trợ
  - Vòng đời Fragment
  - Giao diện XML
  - Lưu trữ Fragment (Back Stack)
2. Xây dựng và sử dụng Fragment



### 1.1 Fragment & Phiên bản hỗ trợ



- ❑ **Fragment:** đối tượng được nhúng trong Activity, cho phép thực hiện nhận tương tác, có vòng đời riêng và thực hiện trao đổi thông tin với Activity và các Fragment khác.
- ❑ **Fragment API:** được thêm vào từ phiên bản Android HoneyComb (API 11).
  - Package: android.app.Fragment
- ❑ **Phiên bản hỗ trợ:**
  - Package: android.support.v4.app.Fragment (API 4 – Android 1.6)





## 1.2 Vòng đời Fragment

- ❑ **Vòng đời Fragment bao gồm các hàm quản lý trạng thái của Activity và các hàm quản lý trạng thái riêng của Fragment, bao gồm:**
  - onAttach(Activity)
  - onCreate(Bundle)
  - onCreateView(LayoutInflater, ViewGroup, Bundle)
  - onActivityCreated(Bundle)
  - onViewStateRestored(Bundle)
  - onStart()
  - onResume()
  - onPause()
  - onStop()
  - onDestroyView()
  - onDestroy()
  - onDetach()



## 1.3 Giao diện

- ❑ **Giao diện Fragment có thể được khởi tạo trực tiếp thông qua thẻ <fragment> trong XML.**
  - Các thuộc tính quan trọng:
    - class: <Class extends Fragment> (Pakage Name + Class Name)
    - name: <Class extends Fragment> (Pakage Name + Class Name)
    - id: Identifier Resource
    - tag: String Resource
    - layout\_width – layout\_height: Dimen
    - layout\_weight: Integer
    - lauout\_gravity: Gravity
    - layout\_margin<Postion>: Dimen



## 1.3 Giao diện



### ❑ Ví dụ: giao diện Fragment trong XML.

- fragment\_layout.xml:

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    class="com.htsi.demofragment.ImageFragment"
    android:id="@+id/imageFragment"
    android:tag="ImageFramgent"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="@dimen/marginFragment" />
```

- Sử dụng trong MainActivity.java như một layout bình thường:

```
setContentView(R.layout.fragment_layout);
```



## 1.4 Lưu trữ Fragment (BackStack)



### ❑ Việc lưu trữ Fragment bao gồm hai giai đoạn:

- Lưu trữ trạng thái Fragment thông qua biến Bundle:
  - Truy xuất biến Bundle trong hàm **onSaveInstanceState**
  - Truyền các thông tin cần lưu trữ vào biến Bundle
- Đưa Fragment vào Stack:
  - Thực hiện khai báo Fragment
  - Cho phép nhúng Fragment vào Activity
  - Dùng phương thức **addToBackStack(String Tag)** trong đối tượng quản lý Fragment để đưa Fragment vào Stack.

### ❑ Có thể truy xuất lại đối tượng Fragment thông qua **Tag**, hoặc khi phím “Back” trên thiết bị được nhấn.



## Nội dung



1. Các khái niệm cơ bản
2. Xây dựng Fragment



## 2. Xây dựng và sử dụng Fragment



- Thực hiện xây dựng Fragment:
  - Khai báo lớp kế thừa từ lớp Fragment
  - Gọi phương thức onCreateView thực hiện tạo giao diện cho Fragment.

### Ví dụ về tạo lớp Fragment và giao diện:

```
public class ImageFragment extends Fragment {
    public View onCreateView(LayoutInflater inflater, ViewGroup parent, Bundle b )
    {
        View rootView = inflater.inflate(R.layout.fragment_main, null);
        return rootView;
    }
}
```



## 2. Xây dựng và sử dụng Fragment



### ❑ Sử dụng Fragment trong Activity, bao gồm hai cách:

- Thực hiện tham chiếu Fragment từ giao diện XML của Activity.
- Khai báo đối tượng FragmentManger, cho phép nhúng Fragment vào Activity từ JavaCode.



## 2. Xây dựng và sử dụng Fragment



### ❑ FragmentManager:

- Đối tượng quản lý Fragment trong Activity.
- Tạo đối tượng FragmentManger:
  - getFragmentManager (API 11 trở lên)
  - getSupportFragmentManager (android.support.vx)

### ❑ FragmentTransaction: đối tượng cho phép thực thi các thao tác quản lý Fragment.

- Tạo đối tượng FragmentTransaction từ FragmentManager:
  - FragmentTransaction ft = getFragmentManager.beginTransaction();



## 2. Xây dựng và sử dụng Fragment



### ❑ FragmentTransaction:

- Một số phương thức quan trọng
  - add()
  - attach() - detach()
  - replace()
  - hide() – show()
  - remove()
  - addToBackStack()
  - setTransition
  - commit()



## 2. Xây dựng và sử dụng Fragment



### ❑ Ví dụ thao tác thêm Fragment vào Activity:

```
// Tạo đối tượng quản lý
FragmentManager manager = getFragmentManager();

// Tạo đối tượng thực hiện phiên tương tác
FragmentTransaction transaction = manager.beginTransaction();

// Tạo đối tượng Fragment
ImageFragment fragment = new ImageFragment();

// Thiết lập chế độ chuyển hoạt
transaction.setTransition(FragmentTransition.TRANSIT_ENTER_MASK);

// Thực hiện thêm Fragment vào Activity
transaction.add(R.id.container, fragment);

// Đưa phiên tương tác vào Stack
transaction.addToBackStack(null);

// Xác nhận tương tác
transaction.commit();
```



## Thảo luận



cuu duong than cong . com

Lập trình Android (2014) – Bài 9. Fragment

14

cuu duong than cong . com